(19) Europäisches Patentamt
European Patent Office
Office européen des brevets

(11) Publication number : **0 631 247 A2**

(12) EUROPEAN PATENT APPLICATION

(21) Application number : **94304000.6**

(51) Int. Cl.[5] : **G06F 15/44**

(22) Date of filing : **03.06.94**

(30) Priority : **24.06.93 US 80836**

(43) Date of publication of application :
28.12.94 Bulletin 94/52

(84) Designated Contracting States :
AT BE CH DE DK ES FR GB GR IE IT LI LU MC NL PT SE

(71) Applicant : **NINTENDO OF AMERICA INC.**
4820 150th Avenue, N.E.
Redmond, Washington 98052 (US)

(72) Inventor : **McCarten, David J.**
14447 89th Place N.E.
Bothell, Washington 98011 (US)
Inventor : **Smith, Darren C.**
5029 148th Avenue, N.E.
Bellevue Washington 98007 (US)
Inventor : **Nishizawa, Kenji**
13023 S.E. 69th Street
Bellevue Washington 98006 (US)
Inventor : **Ravanpey, Ramin**
P.O. Box 45602
Seattle, Washington 98145 (US)

(74) Representative : **Smith, Norman Ian et al**
F.J. CLEVELAND & COMPANY
40-43 Chancery Lane
London WC2A 1JQ (GB)

(54) Video game and communications system to be used on an airplane accessible via an onboared distributed processing system.

(57) An airline-based video game system includes a multitasking master computer (2), which preferably stores video game and other application programs on its hard disk. The master computer is coupled to a set of airplane zone control computers ADBI to ADBN which also perform conventional cabin management tasks. The zone control computers receive data from the master control computer and couple data to identified seat controlling processing units (SEBs). Each SEB receives data from, and couples data to, a set of unique seat display units which are associated with each seat in the airplane. The system downloads application software to the seat display units from the master computer. After receipt of a downloading request, the master computer responds by setting up an application program transmission for generating the display menu which appears on each SDU. The initial applications program downloading results in a menu display at every passenger seat which initiated a request. The applications program is then coupled to each SDU for display on its liquid crystal display screen. The display menu advantageously permits each passenger to select between various operating modes including : movies, games, shopping, survey forms, language selection, communication/data processing services. Communication or data processing services permit selection of in-flight phone services, word processing services, and facsimile services. If a user opts for video game play, then the available game titles and/or descriptions thereof are displayed. The SDU includes interface processors and associated hardware and software which enable high speed downloading operations to be efficiently performed. The master computer initiates a high speed video game program downloading process to enable the user to play the selected video game.

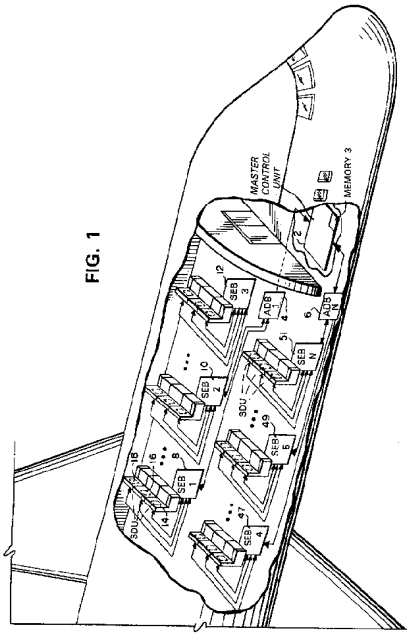


FIG. 1

EP 0 631 247 A2

FIELD OF THE INVENTION

This invention relates generally to a digital communications and entertainment/video game system. More particularly, the invention relates to an airline-based distributed processing video game and communications system associated with substantially every seat in an airplane.

BACKGROUND AND SUMMARY OF THE INVENTION

For the vast majority of airplane passengers, travel time is largely filled with passive activities such as reading magazines, newspapers, or watching a movie provided by an airline during a lengthy flight. Individual passengers who wish to participate in video game play have brought their own portable video game units onto the plane, such as the Nintendo Gameboy product. Similarly, passengers who desire to use a data processor during travel have had to use their own portable laptop computer.

At least one airline has provided limited video game services to first class passengers. This system, however, is believed to be a centrally controlled system in which video game programs are executed by a master computer.

The present invention is directed to a video game/communications system for providing any passenger in the airplane an opportunity to actively participate in video game play or to use other data processing/communication services accessible via an onboard distributed processing communications system.

In an exemplary embodiment of the present invention, a multi-tasking master computer, which preferably stores video game and other application programs on its hard disk, downloads programs which are ultimately downloaded to passengers' seat display units for execution. The master computer, in the exemplary embodiment, is coupled to a set of airplane zone control computers (hereinafter identified as ADBs) which also perform conventional cabin management tasks. The zone control computers receive data from the master control computer and couple data to identified seat controlling processing units (hereinafter identified as SEBs). Each SEB receives data from, and couples data to, a set of unique seat display units which are associated with each seat in the airplane.

In an exemplary embodiment of the present invention, seat display units (SDU) are incorporated into the seat back of each passenger seat (except, of course, of the last row of the airplane). Alternatively, the seat display units may be embodied in a rotatable structure installed in a seat arm rest.

The system downloads application software to the seat display units from the master computer. After receipt of a downloading request, the master computer responds by setting up an application program transmission for generating the display menu which appears on each SDU. The initial applications program downloading results in a menu display at every passenger seat which initiated a request. The applications program is then coupled to each SDU for display on its liquid crystal display screen.

In accordance with a preferred embodiment of the present invention, the display menu advantageously permits each passenger to select between various operating modes including: movies, games, shopping, survey forms, language selection, communication/data processing services. In an exemplary embodiment of the present invention, icons are displayed on the screen to permit a passenger to select any one of the different modes of operation. Communication or data processing services permit selection of in-flight phone services, word processing services, and facsimile services.

If a user opts for video game play, then the available game titles and/or descriptions thereof are displayed. The SDU is itself a multiprocessor system which permits a wide range of video game programs to be executed by a video game computer system. It includes interface processors and associated hardware and software which enable high speed downloading operations to be efficiently performed. The master computer initiates a high speed video game program downloading process to enable the user to play the selected video game.

The above and other features and advantages of the invention and the manner of realizing them will become more apparent, and the invention itself will best be understood, from a study of the following detailed description and the appended claims, with reference to the attached drawings showing some exemplary embodiments of the invention.

BRIEF DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a general block diagram of the video game/communications system in accordance with an exemplary embodiment of the present invention;

FIGURE 2 is a diagram generally depicting the overall operation of an exemplary embodiment of the present invention;

FIGURES 3A and 3B are flowcharts delineating the sequence of operations involved in downloading information to the seat display units;

FIGURE 4 is a block diagram of a seat display unit in accordance with an exemplary embodiment of the present invention;

FIGURE 5 is a block diagram of the video game processing and storage system in an exemplary passenger seat display unit;

FIGURE 6 is a detailed block diagram of the

memory board shown in FIGURE 5;

FIGURES 7A through 7H are memory maps showing the memory configuration in various modes of operation;

FIGURE 8 is a block diagram of an exemplary video game processing circuitry in a seat display unit;

FIGURES 9A-9D are flowcharts delineating the sequence of operations in an illustrative boot program; and

FIGURES 10a through 10l are flowcharts delineating the sequence of operations performed by the microcontroller of the memory board of FIGURE 6.

DETAILED DESCRIPTION OF THE DRAWINGS

FIGURE 1 is a block diagram of an exemplary distributed processing communications system as installed in an airplane in accordance with an illustrative embodiment of the present invention. Although the presently preferred embodiment of the present invention is installed in an airplane, it is contemplated that the communications system described herein may be advantageously used in other environments such as in a hotel, or an ocean liner. In such alternative embodiments, the system would be modified in various respects such as, for example, by using the hotel room television as the display device.

The airline-based system of FIGURE 1 includes a master control computer 2. The master control computer 2 is a multi-tasking computer which, for example, may be an IBM 386 computer running an interactive UNIX operating system. The master control computer 2 includes a conventional communications board which, for example, may be a SEALEVEL ADVANCED Communication Board (ACB-II) Part #3061B, which generates synchronous data link control data and which communicates with compatible communications boards installed in other processing modules shown in FIGURE 1. The master control computer 2 preferably stores video game and other programs in its memory 3, which may include a hard disk memory system. Such programs are ultimately downloaded to a passenger's seat display unit (SDU) for execution.

The master control computer 2 in the exemplary embodiment shown in FIGURE 1 is coupled to a set of airplane zone control computers, designated as ADB 1(4) to ADB-N (6). Both the master control computer 2 and each ADB (1 to N), in addition to performing the video game/communications system tasks described in detail herein, also perform conventional cabin management tasks which are not germane to the present invention. For example, each ADB 1 to N preferably performs under the control of the master control computer 2 such conventional cabin management tasks as assigning seat numbers to each seat in its associated zone, and monitoring the status of the control panels associated with each seat. In a large aircraft such as a Boeing 747, the system may use, for example, 6 ADBs associated with 6 different zones in the aircraft.

The ADBs (1 to N) additionally receive data from the master control unit 2 and couple the data to identified processing units ("SEBs" (1 to M), where M is greater than N) associated with a predefined group of seats. As opposed to exercising direct control over each individual seat, the ADB's (4, 6) couple appropriate control signals to one or more SEB's (8, 10, 12, 47, 49, 51). Each SEB (e.g., 8) includes a processor which exercises control over a group of seat display units (e.g., 14, 16, 18) which are preferably associated with every individual seat in the airplane. Each SEB (e.g., 10) may have, for example, 8 serial ports via which an associated on-board processor receives data from an ADB and passes the data on to any other SEB (e.g., 12) to which it is daisy-chained. Additionally, each SEB (e.g., 8) receives data from and couples data to the set of seat display units (e.g., 14, 16, 18) such as are associated with the seats shown in FIGURE 1.

In the presently preferred embodiment, each seat display unit (SDU) is incorporated into the seat back of each passenger seat (except for the last passenger row). Alternatively, the seat display units may be embodied in a rotatable structure installed in a seat arm rest. The SEBs 8, 10, 12 are preferably disposed underneath one of the seats over which it exercises control. Each ADB 4, 6, is preferably installed underneath the floor in, for example, the airplane aisle in its zone. The master control unit 2 may be installed in the belly of the airplane in the first class section.

FIGURE 2 is an exemplary block diagram which depicts the overall operation of the system shown in FIGURE 1 in accordance with an illustrative embodiment of the present invention. As shown in FIGURE 2, immediately after power is turned on (1), an initialization sequence is performed (3). During initialization, each seat display unit (SDU) performs its own initialization routine during which various parameters are set to appropriate default values and serial ports are initialized to the correct baud rate. Additionally, as will be explained in further detail below, a version identification number associated with the boot program executed in each SDU is coupled to a microcontroller 190 described in conjunction with FIGURES 4 and 9. The microcontroller (190), which is preferably embodied on a memory board 102 (see FIGURE 5) in each seat display unit, receives the version identification number and performs input/output interfacing functions for the memory board 102. The master control unit 2 also performs conventional initialization functions including polling each of the ADBs 4, 6, etc., requesting the ADBs to assign seat numbers and to report back such seat assignments together with an

indication that communication has been established with such seats.

After initialization (3), applications software is downloaded to the seat display units 8, 10, 12, etc., via respective ADBs 4, 6, etc (5) and SEBs. The downloading is initiated in response to a downloading request from each SDU coupled to the master control unit 2 via the respective SEB's and ADB's.

After receipt of the downloading request, the master control unit 2 responds by setting up an applications program transmission for generating the display menu which appears on each SDU (7). The initial applications program downloading results in a menu display at every passenger seat SDU 14, 16, 18, etc., that initiated a request. The applications program is coupled to each SDU through an ADB and SEB via an RF channel selected by an on-board tuner (See FIGURE 4) associated with each SDU. The tuner permits the transmitted application program to be executed by the requesting SDU resulting in a menu display on a liquid crystal display screen associated with each seat in the airplane.

In accordance with the presently preferred embodiment of the present invention, the display menu permits the user to select between various operating modes including: movies 9, games 15, shopping 23, survey forms 25, language selection 27, communication/data processing services 35. Initially, the user may be prompted to initially select a language (27) so as to choose between English 29, German 31, Japanese 33, etc. The language selection, in turn, determines the language used with, for example, any subsequent movie selection, etc.

In an exemplary embodiment of the present invention, icons are displayed on the screen to permit the user to select any one of the different modes of operation. If the movie option 9 is selected, a check is made to ensure that the movie is presented in the appropriate language (11). The channel is then changed (13) for receipt of the selected movie via an associated VCR (not shown).

Communication or data processing services 35 may be selected to select between in-flight phone services 43, word processing services 45, and fax services 37. It is contemplated that the seat display units may include a port for receiving a keyboard input which, in one illustrative embodiment of the present invention, is distributed by airline personnel upon request. Alternatively, the system may be expanded to include, in association with at least some seats in the airplane, a keyboard which is rotated into operating position upon selection of word processing 45 or facsimile services 37.

If the passenger selects facsimile services 37, then the user is prompted to begin formulation of a message to transmit 39. After composition of the message, the user enters an end of message or other special control character to indicate that the message may be transmitted (41).

If the user opts for video game play (15), then the available game titles and/or descriptions thereof will be displayed to the user (17). Thereafter, the master control unit 2 initiates a video game program downloading process which is explained in detail below in conjunction with FIGURES 3A and 3B (19). The system then begins executing the game program and the passenger is able to play the selected video game.

The communications system of the present invention also includes a shopping service option (23) in which a wide range of available items may be selected by the user for purchase via credit card. Additionally, the system has the capability of requesting passengers to complete survey forms (25).

The entertainment and data processing services selectable via the communications system of the present invention shown in FIGURE 2 are by way of example only. The present invention contemplates that additional services may be among the selectable options such as books or magazines which may be stored in a mass storage media associated with master control unit 2. Additionally, it is contemplated that a selection of educational programs may be provided in addition to the selectable video games.

FIGURES 3A and 3B are a flowchart delineating in more detail the sequence of operations performed during the downloading operation shown at blocks 5 and 19 of FIGURE 2. As indicated in FIGURE 3A, upon initiation of the download operation (20), execution of the boot program results in the forwarding of a "download type" request to the SDU's interface controller 84 (FIGURE 4) via microcontroller 190 (FIGURE 6). The downloading may be of either an applications program, a specific video game program, or any other program executable on the system. Upon receiving the request regarding the "download type", the SDU interface controller 84 responds by sending instructions and parameters back for use by the boot program including, for example, the file number and other parameters such as may be used in defining an address mapping mode (as is explained in detail below) (24). The SDU controller 84 saves such information in its memory and the controller makes a decision regarding what should be downloaded next. The file number is sent to an SEB and is passed through the microcontroller 190 which saves the file number for the downloading process. Upon receipt of such information, a nonvolatile memory on a memory board (100 in FIGURE 5) is checked by the executing boot program to determine whether a file having a file number matching that just received is stored in a pseudo-static RAM on the memory board (26). If the file does exist in memory, as indicated by the check in block 26, then a checksum verification of the identified area in memory is performed (28). If the checksum test is passed, then program execution is started (32) since the downloading operation has already

been completed (34).

If the checksum test is not passed, or the determination in block 26 reveals that the file does not exist in memory, then the boot program being executed by each game processor requests that the identified file number be downloaded and forwards such a request to an SEB. The SEB forwards it to the master control unit (36). The SDU receives an acknowledgement of the request for a file number to be downloaded (38) after which downloading is scheduled to commence.

The boot program, as indicated at block 40, next checks to determine whether downloaded data has been received. If not, a further check is made to determine whether a timer has expired (42). If the timer has not expired, then the routine branches back to block 40 where a check is again made for downloaded data. If the timer expires before downloading has commenced, the downloading branches back to node B of FIGURE 3A, where the downloading process begins again. If data has been received, the boot program reads the first byte of downloaded data (44) and treats the first byte of data as identifying, for example, the number of memory banks expected (46). In accordance with an illustrative embodiment of the present invention, the order in which such "configuration" data must be received may function as a "security" mechanism which permits only authorized programs to be downloaded.

As indicated in FIGURE 3B, at block 48, the next byte is then read. A check of the next byte is made to determine whether it identifies the proper bank number (50). If a proper bank number has not been identified, the download is aborted (52) and the routine branches back to node B to start the process again.

If the proper bank number has been identified, then the downloaded memory bank number is set up (54) and the next two bytes of data are read (56). As indicated in block 58, a check is made to determine whether the two bytes of data read in block 56 identify the correct memory starting address. If the correct starting memory address has not been defined, then the routine branches to block 52 to abort the download. If the proper memory starting address has been identified, as indicated in block 60, the identified starting address is set up. The next two bytes of data are then read (62) identifying the number of bytes in the bank and a check is made in block 64 to determine whether the proper number of bytes have been identified. If not, the routine branches to block 52.

If the proper number of bytes have been identified, then data is read at block 66, which is actual program related data. Thereafter, after each byte is read, the byte is written to memory 68. As indicated at block 70, a check is then made to determine whether all bytes in the bank have been received. If all the bytes have not been received, then the routine branches back to block 66. If all bytes have been received, then a check is next made to determine whether all the identified banks have been received (72). If all banks have not been received, then the routine branches to block 48 for further bank processing.

If all banks have been received, then a checksum is calculated of the downloaded memory contents (74). If the checksum matches, a stored checksum value, the program is executed (80) and the routine ends. If the checksum does not match, then the routine branches to node B in FIGURE 3A where the downloading process begins again. There is a second timer activated when downloading starts. If the entire download process is not finished within a preset time, then the download is aborted, and the process starts from the beginning, i.e., block 22 of FIGURE 3A.

FIGURE 4 is a block diagram of an exemplary seat display unit (SDU) in accordance with the presently preferred embodiment of the present invention. The exemplary seat display unit shown in FIGURE 4 includes a video game/processing board 82 which is described in further detail in the figures below.

The video game processing board 82 is coupled to a duplex RS 232 type communication bus that is coupled to an SEB (e.g., 8, 10, 12) in the airline video game system of FIGURE 1. The video game processing board 82 is also coupled to a video game controller which may be the hand-held controller used in conjunction with the Super Nintendo Entertainment System (SNES) sold by the assignee of the present invention. The video game processing board 82 also receives program instructions and data at high speeds from tuner 86 via the download bus shown in FIGURE 4. The tuner 86, in turn, receives program information and data via an RF video channel from the master control unit 2 which couples program instructions and data to the requesting SEB. The tuner 86 in response to the incoming signal on RF video bus selects an identified channel in a conventional manner and couples a received composite video signal to interface 84. If a user selected a movie, the tuner would be programmed to select a predetermined channel. Alternatively, if a user selects a video game program, then a predetermined data channel is selected for the downloading of program information and data. The tuner 86 is selectively switched by a commercially available interface controller 84. The tuner 86 may be an off-the-shelf tuner which may, for example, be a Phillips FS936E.

The microcontroller interface 84 includes an 8 bit microcontroller which performs interface operations for controlling the LCD display 90 and the tuner 86. The microcontroller may, for example, be a commercially available 8051 microprocessor manufactured by Phillips Electronics. Interface 84 may, for example, supply the color, contrast, brightness, and other control signals for controlling the LCD display 90. The microcontroller interface 84 additionally reads information from the magnetic card reader 88. The micro-

controller interface 84 includes on-board random-access memory (not shown). The microcontroller interface 84 receives the composite video signal from the video game processing board and couples such signals to the LCD display. The microcontroller interface 84 communicates with the video game processing board's on-board microcontroller (MCU 190 in FIGURE 6) via a serial communication link. The communication port coupled to microcontroller 190 which is linked to microcontroller interface 84 is referred to herein as the SDU port of the MCU 190.

The seat display unit also includes a magnetic card reader 88 which is used to read information magnetically stored on the magnetic strip of a user's credit card. The information is read via the interface processor's serial I/O port which couples information to the video game processing board 82 which, in turn, sends this billing information to a SEB, an ADB and to the master control unit 2 for storage. The SDU additionally includes an LCD dot matrix display 90 which may, for example, be a conventional 6" panel Color TFT-LCD AV type display manufactured by Sharp Corporation commercially sold under manufacture model #LQ6NCO2. The LCD display 90 receives a composite video signal from the interface 84 which, in turn, receives the composite video signal from tuner 86. Tuner 86 also provides the LCD display 90 with video control signals such as the signals controlling the horizontal and vertical scanning operations and provides color, brightness, contrast and tint related signals in a manner understood by those skilled in the art.

In operation the video game processing board 82, after power up, informs the interface processor 84 that it is in need of instructions. The interface processor 84 responds to the request from the video game processor board 82 by providing an indication of the type of file which must be downloaded and the associated parameters which must be received. The interface processor 84 is programmed to distinguish between power-on reset and other error induced resets and provides the instructions to the video game processor board 82 to either request a display menu or a particular type of program to be downloaded. The video game processing processor 82 then requests via the RS232 communication bus and its associated SEB, the appropriate downloading operation. The master control unit 2 then downloads the required information via tuner 86 to the appropriate display unit. A composite video signal is passed from video game processing board 82 to microcontroller 84 to LCD display 90.

FIGURE 5 is a general block diagram which shows the significant data and control signals associated with the video game processing board 82 of the SDU. The video game processing board 82 includes a video game computer board 100 and a memory board 102 both of which are described in further detail below. In the presently preferred exemplary embodiment of the present invention, the video game computer board is a compact version of the Super NES video game system (which lacks the RF modulator module that in the commercial SNES couples the processor to a conventional television monitor). The memory board 102 includes storage devices for storing downloaded game program, game character data and other applications program information. The memory board 102 additionally includes a boot read-only memory (ROM) whose boot program executed upon power on determines whether the pseudo static RAM in the memory board 102 contains the expected program information and performs other operations as explained below in conjunction with FIGURES 9A-9C. In the presently preferred embodiment, the memory board also contains a microcontroller 190 shown in FIGURE 6 together with a ZILOG communications controller model number Z85233.

Turning next to the data and control signals which are exchanged between the video game/computer board 100 and the memory board 102, a refresh signal REFRESH is coupled to the memory board 102 to refresh the random access memory (RAM) devices in a manner that will be appreciated by those skilled in the art. The computer board 100 also couples a system clock signal and a 21 MHz clock signal to the memory board. The system clock signal provides the necessary clocking for register functions and for the memory board RAM in a manner understood by those skilled in the art. The system clock preferably permits the clocking rate to be selectable to at least a limited extent. The ROMSEL and the RAMSEL signals are generated by the video game computer board and are used as chip enable signals which are processed by decoding logic in the memory board to select the appropriate memory at the appropriate time. As shown in FIGURE 5, various power lines and bidirectional control lines are also coupled to the memory board and the video game computer board.

Among the control signals coupled to the video game computer board 100 are video game control signals generated by a player hand-held controller. These signals are coupled to the video game computer board 100 via a microcontroller on the memory board 102. In a preferred embodiment of the present invention, the signals generated by a conventional SNES type game controller are latched by a logic circuitry within the memory board 102. In accordance with the present exemplary embodiment, the controller signals are first routed through a SEB associated with the SDU and are coupled to the microcontroller which latches the player input signals and couples these signals to the video game computer via the controller lines shown in FIGURE 5. In the memory board 102, two 8-bit latches are used to provide 16 bits of player controller information to the video game computer board 100.

The video game computer board has 24 address lines CA0-CA23 which are coupled to the memory board 102 and used to address the memory devices embodied therein. Additionally, 8 data lines are used for exchanging data between the memory board 102 and the computer board 100. The memory board 102, as will be explained further below, includes a pseudo static RAM for storing program information which, like other memory in memory board 102, is coupled to the address data lines. The video game computer board 100 also includes additional control lines for coupling read or write signals to the memory devices on the memory board 102.

The memory board 102 also includes an IRESET line for providing an externally generated reset signal to the video game computer board 100. IRESET is used when the system needs to be reset due to communication problems which may develop from time to time on an airplane or when communication must be terminated due to other higher priority communications. The RESET line shown in FIGURE 5 is used to reset the elements embodied in the memory board upon power-up to permit voltage levels to settle. The computer and memory board are also interconnected via an 8 bit address bus PA0-PA7 which permits addressing of registers to be described below which are located in a particular CPU address space.

The memory board 102 receives a high speed synchronous serial input which includes program information downloaded from the master control unit via tuner 86 such as shown in FIGURE 4. Such synchronous serial input is received at, for example, one Mbaud which may include game program or application program information which has been converted into the appropriate format via the ZILOG Z85233 communications controller. The memory board also includes an asynchronous serial input port which receives input at 9600 baud including game controller data that is received from an SEB. Additionally, a further asynchronous input port is included which receives information from the microcontroller interface 84 at, in an exemplary embodiment, 9600 baud. The memory board 102 receives a 5 volt power signal. Additionally, the video game computer board 100 outputs video signals which are coupled to the LCD display 90 via the microcontroller interface 84 and additionally, outputs left and right channel audio signals.

FIGURE 6 is a block diagram of the circuitry embodied on the memory board 102 shown in FIGURE 5. The memory board 102 includes decoder logic 150, 152, 154, 156 which may include associated registers 158, 160, 162, 164 and may, for example, be implemented programmable array logic (PAL). The decode logic 150-156 performs decoding and register loading related functions as will be explained in detail below.

Associated with each decode logic 150, 152 154, 156, is a single bit register. Register 158 is identified as the speed register. Register 160 is identified as the Zbank. Register 162 is identified as the map mode register and register 164 is identified as the boot/run register. The function of the registers 158-164 are explained further below. The bits stored in the respective registers are input to a pseudo RAM (PSRAM) controller 166 which, in turn, selects in accordance with the state of the output from registers 158-164, the pseudo RAM 174 address mapping mode. The address mapping functions implemented by the PSRAM controller 166 permits diverse games using different address mapping modes to be executed using the same memory board hardware.

The pseudo-static RAM controller 166 in addition to performing address mapping functions also provides a pseudo-static RAM output enable signal for read function, a write enable signal, and generates the refresh signal required for the pseudo static RAM 174. The pseudo static RAM controller 166 receives address data from the Super NES address lines (as shown in FIGURE 5). This address data is interpreted dependent upon the state of registers 158-164, which, in turn, are set in response to the address signals input from decode logic 150-156 via address lines as PA0-PA7. In the illustrative embodiment, the pseudo-static RAM 174 is preferably a 2 Mbyte RAM which is also responsive to PSRAM controller signals as shown in FIGURE 6.

The registers 162, 164 are also coupled to static RAM controller 168 which controls access to RAM 176 by generating a chip select signal based upon output that is received from registers 164, 162. The static RAM 176 is addressed via address signals on the Super NES address bus and is responsive to the Super NES read and write control signals as shown in FIGURE 6.

The registers 160, 162 and 164 are also coupled to a non-volatile RAM controller 170 which generates a chip select signal for non-volatile RAM 178. Non-volatile RAM 178 is addressed from the SNES address bus and receives write control signal and read control via chip enable as shown in FIGURE 6. The contents of boot/run register 164 as well as SNES reset and ROM select signals are coupled to EPROM controller 172 which generates a chip select signal at the appropriate time to read the EPROM 180. The EPROM controller 172 receives an address from the Super NES address bus. The EPROM may be written in response to an SNES write control signal. Each of the pseudo-static RAM 174, SRAM 176, non-volatile RAM 178 and boot ROM 180 is coupled to the SNES address and data buses.

The pseudo-static RAM 174 stores either a downloaded game program or the downloaded applications programs as described above. The static RAM 176 stores various types of game parameter information and operates as a scratch pad memory. The non-volatile RAM 178 stores information generated by an applications program and information relating to the

status of the pseudo-static RAM 174 including information identifying the kind of data stored in the pseudo-static RAM 174.

The memory board 102 also includes in the present exemplary embodiment an interface microcontroller 190 which may, for example, be a Hitachi H8/325 microcontroller. The microcontroller 190 performs functions which are described in detail below.

Memory board 102 additionally includes a control decoder 182 that is coupled to the SNES address lines. In response to signals received on the SNES address lines, control decoder 182 couples a "data ready" signal to microcontroller 190, a "read" signal to first-in first-out (FIFO) buffer 184, provides a "data shift in" signal to latch 188 (which receives data from the SNES data lines) which, in turn, shifts data out to microcontroller 190. The FIFO 184 receives high speed downloaded information from microcontroller 190 and stores such data in response to the "write" signal generated by MCU 190. The control decoder 182, in response to a read control signal on its input address lines triggers a read operation from FIFO 184. If there is no data available in FIFO 184 upon request, a "data not ready" signal is generated by FIFO 184 which is coupled to control decoder 182 and to the SNES data lines. To write data to MCU 190, the SNES processor checks the "busy" line which indicates if MCU 190 can receive data. If MCU 190 can receive data, then one byte is shifted in latch 188, which, in turn, activates the "Busy" signal by sending a "Input Strobe" signal. If MCU 190 cannot receive data, SNES continues to check the "busy" signal.

The microcontroller 190 additionally controls a ZILOG serial communications controller 192 which is coupled to the tuner 86 shown in FIGURE 4 for receiving high speed downloaded program instructions and data. The downloaded program instructions and data are coupled to the ZILOG serial communications controller 192 via voltage level shifter 194. The high speed downloaded data from the tuner 86 has a logic level of 0 to 1 volts. Level shifter 194 is a conventional level shifter which converts the 0 to 1 volt data to 0 to 5 volts.

The memory board 102 also includes a halt controller 196 which is coupled to microcontroller 190. The halt controller 196 is designed to couple a halt signal to the video game computer. The halt signal may be generated to halt game play after a predetermined time period, e.g., 1 hour after initiation, so that the user may be prompted to request further playing time and to pay for such time. Additionally, the halt controller may be programmed to be responsive to public address announcements or other events on the airplane deemed to be events which should trigger the halt condition. The halt controller 196 ensures that the halt does not take place at any arbitrary time, but requires halting in sync with a memory refresh operation to avoid losing stored data and the system clock signal. Halting the Super NES may be desirable when higher priority tasks must be performed or if, for example, some other reason exists for halting the video game computer such as during communications or power problems.

As previously indicated, memory board 102 couples game controller data to the video game computer board through controller input lines. Such controller data is coupled to microcontroller 190 via an asynchronous serial port (the SEB port) via an SEB. The controller data is output to the SNES through latch 186 whose output is coupled to a controller data lines as shown in FIGURE 5.

The microcontroller 190 also includes a reset output line which is used to reset the video game computer to, for example, recover from a temporary drop in voltage level in the airplane or any other electrical disturbance associated with the relatively hostile airplane environment. The boot ROM program checks to determine whether it is executing as a result of such error condition.

FIGURES 7A-7H depict various configurations of the video game computer address space. The memory configuration accessible by the video game computer CPU is defined by the information stored in registers 158, 160, 162 and 164, as exemplified in FIGURES 7A-7H.

FIGURES 7A and 7D show two exemplary memory configurations after the power is initially turned on. As shown in FIGURES 7A and 7D the boot ROM program is initially executed which is accessed by accessing memory bank 00. During this time period no application program is running, as indicated by the boot/RUN register storing a logical "0". The boot/run bit switches between the boot ROM being mapped to a location for the video game computer to execute (boot/run = 0) or the contents of the pseudo-static RAM being mapped to a location position for the video game computer to execute (boot/run = 1). As shown in FIGURES 7A and 7D, the contents of the "speed" and "Zbank" registers have no effect on this memory configuration (as indicated by the "X" or "don't care" condition). The registers 158-164 are set in response to the selection of a game program prior to downloading a specific game program.

FIGURE 7A reflects the memory configuration which is the standard configuration where the SNES video game computer would normally begin executing out of a game cartridge ROM. In this configuration, the boot ROM embodied on the memory board 102 is mapped in place of the game cartridge ROM. In the initial power up mapping shown in FIGURE 7A, the non-volatile RAM, (NVRAM) is accessible by the video game computer to enable the boot ROM program to provide a check of the last status of the system prior to being powered up (which is stored in a non volatile RAM 178). As shown in FIGURE 7A during the boot program execution, the pseudo-static RAM 174,

the static RAM 176, and the NVRAM 178, are accessible by the CPU.

FIGURES 7B, 7C, and 7E define memory three standard address space configurations associated with a wide range of Super Nintendo Entertainment Systems games. As indicated in 7B, 7C, and 7E, the contents of the boot/run register is "1" indicating that the game program is executing.

The memory mapping modes shown in FIGURE 7B and 7C differ in that in FIGURE 7C an image of the pseudo static RAM locations appears in predetermined lower address memory banks and higher memory address banks. As shown in FIGURES 7B and 7C, the associated register states differ based upon the contents of the "speed" register. As previously indicated, the video game computer system clock generates signals at two different rates. In an exemplary embodiment of the present invention, the high speed clock rate is used to execute programs (stored in an image of lower order PSRAM memory banks) out of higher order address banks.

The Zbank register setting controls the ability of an applications program to, upon completion of execution, permit the video game computer to access the boot ROM to permit down loading of a game program. A game program, however, cannot be permitted to access the boot ROM. The state of the Zbank register is fed back to the decoding logic to indicate that the application program is running. The map mode bit defines in part the address mapping mode to be selected.

If a video game program is running, the state of the Zbank register precludes modification of the speed, Zbank, map mode, or boot run registers. However, if an application program is running, the state of these registers may be later modified.

The memory board 102 shown in FIGURE 6 operates as follows. When power is initially turned on, video computer board's CPU is represented as shown FIGURE 7A and the program stored in the boot ROM is executed. The boot ROM program execution (which is explained in further detail below) requests the video game computer CPU to write data appearing on the video game computer data lines into latch 188 in response to "data shift in" control signals generated by control decoder 182 in response to signals on the SNES address lines. The microcontroller 190 is informed that data is ready via its "Input Strobe" control input. The MCU 190 reads the data stored in latch 188 and outputs the data to the SNES video game processor, the SDU or the SEB 2.

Data which is to be coupled to the SNES is transmitted loaded by MCU 190 into FIFO 184. The SNES video game computer, when executing programs stored in the boot ROM, monitors FIFO 184 for the presence of a status flag when data is available. The SNES then couples a control signal on its SNES address lines which are decoded by control decoder 182 to generate a "read" signal which is coupled to FIFO 184 which initiates the read out of information from the SNES data lines.

The boot ROM program may then, in response to the read data, initiate the downloading of an applications program, a game identifier which uniquely indicates the game which has been selected and/or mapping mode register indicia that is to be loaded into registers 158-164. The first time the boot ROM program is executed the applications program downloading is initiated. As the application program is downloaded, the boot ROM monitors the FIFO 184 status flag for the presence of information which may be read. The applications program itself is downloaded via a high speed download link through level shifter 194 to the ZILOG serial communication port controller 192 which, in response to control signals from the microcontroller 190, couples data to the microcontroller which, in turn, loads the data and/or instructions to FIFO 184.

When the applications program is downloaded from the master control unit 2, after being buffered in the FIFO 184, it is loaded into the pseudo-static RAM 174 via the SNES data bus. After the applications program is downloaded, game related parameter data is loaded and stored in the nonvolatile RAM 178. In this fashion, the applications program is able to access during execution the contents of the non-volatile RAM 178 to enable display of such information as the particular games and/or educational programs that are available. When the applications program has been successfully downloaded, the contents of the z bank register is set to "1" and the video game computer memory address space is organized as shown in FIGURE 7F-7H.

During the execution of the application program, the user makes the desired menu selections. The menu selections result in the loading of data into latch 188 in FIGURE 6. The "Input Strobe" signal is then sent to the microcontroller 190 which triggers the reading of data from latch 188 to result in the menu selection data being sent to the master control unit 2 through an SEB and ADB. The ultimate destination of the data depends upon the user's menu selection which may indicate the need to download a particular video game program, or the user's selection of a movie, shopping or some other mode.

If a game is selected, player control data indicating, for example, the movement of a moving object, e.g., Super Mario, is coupled to the MCU 190 from a SEB which reads the player controller information to latch 186. The player control data is then coupled to the video game controller via SNES controller data lines. The player controller data may indicate, left, right, up, down directional movement of a moving object, or control signals generated by the "A", "B", etc., control buttons on a standard SNES controller.

During execution of the program stored in the

boot ROM it is permissible for the contents of registers 158-164 to be changed. Addresses appearing on address lines PA0-PA7 are used to uniquely set or reset each of the four registers 158, 160, 162, and 164. PAL decode logic 150, 152, 154 and 156 prevents associated registers from being set or reset during the time period when the system precludes such a modification. For example, during execution of a game, the decode logic 150 will not permit the clock rate of a game to change.

FIGURE 8 is a simplified block diagram of an exemplary computer/video game processing system which may be used in conjunction with the present invention. In accordance with the present exemplary embodiment, the computer board may, for example, be the 16 bit video game system commercially sold by Nintendo of America Inc., as the Super Nintendo Entertainment System (Super NES). The Super NES is described in part in U.S. application Serial No. 07/651,265, entitled "Video processing apparatus", which was filed on April 10, 1991 and U.S. application Serial No. 07/749,530, entitled "Direct Memory Access Apparatus and External Storage Device Used Therein", and filed on August 26, 1991 and U.S. application serial number 07/793,735, filed November 19, 1991, entitled Mosaic picture Display Apparatus and External Unit Used Therefore." These applications are expressly incorporated herein by reference. It should be understood, however, that the present invention is not limited to Super NES related applications and may be used in conjunction with other video game/data processing systems or other non-video game information processing apparatus. Thus, the references throughout the specification to Super NES (SNES) should not limit the scope of the present invention to Super NES related applications or systems having a block diagram such as shown in FIGURE 8. The Super NES preferably is modified in a number of respects such as those described herein. The RF modulator circuitry contained in a conventional Super NES system is not embodied in the present exemplary embodiment since the system is not being coupled to a color television monitor. As shown in FIGURE 8, the video game computer board 100 is coupled to the memory board 102 which was described in detail in FIGURE 6. The host CPU 220 and the other hardware components on board 100, as indicated above, are representative of the Super NES commercially sold by Nintendo of America.

The host CPU 220 is a 16 bit CPU which may, for example, be a 65816 compatible microprocessor. CPU 220 is coupled to a working RAM 226 which may, for example, include 128K bytes of storage. The CPU 220 is coupled to a picture processing unit 222 (which is described in detail in the '265, '530 and '735 applications) which, in turn, is coupled to a video RAM 228. The CPU 220 can only access the video RAM 228 via the PPU 222 at times other than active line scan, when the PPU 222 is accessing video RAM 228. PPU 222 generates a video signal which is coupled to the LCD display 90 shown in FIGURE 4. CPU 220 is also coupled to an audio processing unit APU 224 which is coupled to its working RAM 230. The APU 224, which may comprise a commercially available sound chip, generates the sounds associated with the video game stored in the pseudo-static RAM 174 on memory board 102. Host CPU 220 can only access the working RAM 230 via APU 224.

The video RAM 228 in the Super NES is loaded with appropriate character data stored in the pseudo-static RAM 174 (which stores not only the game program but also the character data used during game play). Any moving object or background characters to be displayed are resident in video RAM 228 before display.

The program storing pseudo-static RAM 174 is accessed by the host CPU 220 via address busses and data busses which are generally shown in FIGURE 8. The PPU 222 is connected to the memory board via shared host CPU data and address busses and via connector 234 to provide a path for PPU data and control signals to be coupled to the memory board. The APU 224 is connected to the memory board via shared host CPU busses and audio bus 232.

As previously described and as indicated in FIGURE 8, the Super NES generates a variety of control signals. When the Super NES CPU 220 needs to access pseudo-static RAM 174, it generates control signal ROMSEL. To initiate a memory refresh, the Super NES generates a refresh signal RFSH. The host CPU 220 additionally generates read and write signals. System timing signals are generated from timing chain circuitry 210 within the video game processing board 100. A power-on reset signal is also generated within the video game computer board 100 and is coupled to the memory board 102. Other control signals shown in FIGURE 8 which are unique to the present airline application implementation have been previously described such as the "halt" control signal and the "IRESET" signal described in conjunction with FIGURES 5 and 6. A more complete portrayal of the signals exchanged between the memory board 102 and the video game/computer board 100 is shown in FIGURE 5.

FIGURES 9A-9C are flowcharts which delineate the sequence of operations performed by the ROM boot program. When the boot program begins executing, the CPU 220, PPU 222, registers and ports associated with the video game/computer board 100 are initialized (251). After initialization, a display screen is generated on the LCD display 90 (252). The initial display screen alerts the user that a program or data downloading operation is occurring. A screen display is preferably generated to maintain the user's interest for the short period of time that the downloading op-

eration takes place. Thereafter, a "download type file" request is made to the SDU via the microcontroller interface 84.

The response to the "download type file" request identifies whether an applications program, game program, or data file is to be downloaded. The response also identifies whether a forced download is to occur. In a forced download, the download is implemented regardless of the current memory contents. In a non-forced download, a check is made of memory to determine whether the file to be downloaded information already resides in memory. If the information is not resident in the memory, then the request is coupled to an SEB for downloading.

In accordance with block 254, a check is made of the response received to the "download type file" request to determine whether a game is to be downloaded. If a game has been requested, the routine branches to block 255 shown in FIGURE 9C. At block 255, a determination is initially made whether a forced download is to occur. If a forced download is not to occur, then a check is made of the non-volatile RAM memory 178 (256 and 257) to determine whether the file in question had already been downloaded. If the check at block 257 indicates that the game program file is already in memory, then the routine branches to block 279 of FIGURE 9B where the working RAM is cleared. The routine then jumps to the game program which is resident in memory.

If the game program file is not resident in memory or if a forced download was determined to occur in block 255, then the downloading process starts (258). The downloading then process is performed (259) until completion (260). The downloading process is performed as described above in conjunction with FIGURES 3A and 3B.

A check is then made to determine whether errors occurred during the downloading process (261). If an error is detected, then the routine branches to block 253 in FIGURE 9A where the downloading process begins again. If no errors are detected, then the downloading process is completed (262) and the routine branches back to block 257, where a check is made to determine that the game program is now resident in memory. The check in block 257 should then reveal that the file is resident in memory and the routine branches to block 279 where the working RAM is cleared and the program execution begins (280) as shown in FIGURE 9B.

Turning back to FIGURE 9A, if the check at block 254 reveals that a game program is not requested, then a check is made in block 263 to determine if a data file has been requested. If a data file has been requested, the routine starts the downloading process for the data file (264). After requesting the data file to be downloaded, the system waits for receipt of the first byte of data (265). The data is then received (266) in accordance with the methodology described above in conjunction FIGURE 3A and 3B. A check is next made to determine whether any errors occurred in the downloading process (267). If an error is detected in the downloading process (267), then an error status flag is set (269). If no error is detected, then the error status flag indicates a logic "O" condition (268). The error status flag setting completes the downloading process (270) and the routine branches back to block 253 to begin the downloading process again by requesting a "download type" file to determine what needs to be done next with the data that has been downloaded.

If the checks in block 254 and 263 reveal that neither a game program nor data has been requested, then in the present exemplary embodiment, an applications program has been requested. A check is made at block 271 to determine whether a forced download is to occur. If a forced download has been requested, then the routine branches to block 274 to initiate the applications program downloading process which is performed in block 276 and completed in block 277 using the methodology described above in FIGURES 3A and 3B. The routine then branches to block 278 in FIGURE 9B where an error check is made on the downloading process. If no error is detected, download is completed (278A), the working RAM is cleared (279) and the applications program is executed (280). If an error is detected then the routine branches back to block 253 to begin the downloading process again.

If the check in block 271 reveals that no forced download was initiated, then the nonvolatile RAM 178 is checked (272). A determination is made based on the nonvolatile RAM 178 contents as to whether the applications program is resident in storage. If not, then the process described above, beginning at block 274 begins to initiate downloading. If the application program is resident in storage as determined at block 273, then the routine branches to block 279, where the working RAM is cleared and program execution begins (280).

FIGURE 9D describes the sequence of operations which occur during boot ROM non-maskable interrupt processing, which is the only interrupt that may occur in the boot ROM program. As indicated in block 281, a "wait" timer is incremented which is used to control various timers, such as the timer which controls the allowable waiting time for a download to start. Next, as indicated in block 282, the display is updated with clock information (e.g., hand movement) to indicate on-going operation of the system. Any additional messages which need to be displayed are then displayed (283). In an exemplary embodiment of the present invention, during the downloading process, a downloading indicator is changed (284) to indicate the degree of download completion. After the interrupt routine is exited, the routine goes back to the main boot routine shown in FIGURES 9A through 9C.

FIGURES 10A-10I is a flowchart which delineates the sequence of operations performed by the microcontroller (MCU) 190 shown in FIGURE 6. The microcontroller 190 executes software whcih is interrupt driven and which continuously check various status flags. Depending upon the state of the status flags microcontroller 190 proceeds to either send data or receive data on a byte-by-byte basis.

Turning to FIGURE 10A, a series of initialization steps are performed to begin the controller's main program loop. Once power is turned on, as indicated in block 300, the interrupts associated with the microcontroller 190 are disabled. Thus, if serial data is received on the controller's input port during this time period, the initialization is not be interrupted. Thereafter, variables are initialized which are utilized to determine what action is to be taken (302). As indicated at block 304, data direction registers associated with the MCU 190 are initialized to determine whether a particular pin is to operate as an input pin or output pin. Thereafter, the MCU serial ports are initialized to set the appropriate baud rates, and the number of start, stop and parity bits (306). Thereafter, the baud rate to be associated with the serial communication controller (192) ports are initialized (308). The microcontroller 192 software then resets the FIFO 184 (310) and resets the Super NES CPU 220 and PPU 222 (312) to ensure that the Super NES begins executing out of the boot ROM 180. Thereafter, buffers which couple the Super NES player controller data to the SNES are initialized (314). As indicated in step 316, the Super NES is then halted and MCU 190 interrupts are enabled (318). The built-in timer is then initialized (320). This timer is used in the MCU 190 to trigger time-controlled interrupts which are initialized when the timer, for example, counts down to zero from an initial value. The initialization process is completed upon the release of the SNES from its halt state (322).

Turning to FIGURE 10B, a significant portion of the MCU 190 program is dedicated to checking the state of various status flags and performing operations in response thereto. One such status flag determines whether the boot "version ID" has been requested (324). Thereafter, if the flag indicates that there has been a version ID request, then a boot version ID Request is sent to the SNES (326). If the version ID request flag has not been set or if the data has been sent in block 326, the routine progresses to block 328, where a check is made to see whether there is SEB data to be transmitted. If so, then the SEB data is sent (330).

If no SEB data is to be transmitted or if the SEB data has already been sent, the routine proceeds to block 332, where a flag is checked to determine whether SDU data is to be transmitted. If SDU data is to be transmitted, as indicated in block 334, the SDU data is transmitted. If the SDU data is not to be transmitted, or if the data has been sent, then the routine enters a "data receive" mode and checks to see if SEB data has been read and not processed (336). The determination in block 336 indicates whether data has been received at the serial port and stored in a buffer of MCU 190.

If there is data in the buffer, then a check is made to determine whether the SEB port is busy (338). If the SEB port is not busy, as indicated in block 340, the SEB data is processed in one byte increments. If the SEB port is busy, if the check at block 336 yields a "NO" response, or if the data has already been processed, the routine branches to block 342 which is checked to determine whether SDU data has been read and not processed. If SDU data has been read and not processed, a check is made to determine whether the SDU port is busy (344). If not, then the SEB data is processed one byte at a time (346).

If the SDU data has not been read and not processed (342), if the SDU port is busy, or if the SEB data has been processed, then the routine branches to block 348, where a flag is checked to determine whether the SNES data has been read and not processed. If the SNES data has been read and not processed, then a check is made to determine whether the SNES is busy (350). If the SNES is not busy, then the SNES data is processed, one byte at a time (352). If the check at 348 reveals that the SNES data has not been read and not processed, if the SNES is busy, or if the SNES data has been processed, then the routine branches back to FIGURE 10B at block 324.

In essence, the MCU 190 time shares processing time between the SEB data, SDU data and SNES data for processing one byte at a time. Data is sent from the MCU 190 out to other system components one byte at a time. The MCU 190 in this fashion services all units at the same time by appropriately routing information which is to be transmitted and/or received.

FIGURE 10D shows the operations involved in sending SNS data according to block 326. A check is first made to determine whether FIFO 184 is waiting for high speed download as indicated in block 354. If FIFO 184 is not waiting, then a request for boot version ID is sent (356). If FIFO 184 is waiting, then the routine returns to the main program.

FIGURE 10E is a flowchart which indicates in further detail the manner in which both the "send SEB" and "send SDU" data operations of blocks 330 and 334 of FIGURE 10B are carried out. The operations for these "send SEB" and "send DSU" data transmissions are substantially the same and accordingly both are described in the flowchart of FIGURE 10E. These transmission steps takes place when data is to be sent out of the MCU 190 SEB serial port or the MCU 190 SDU serial port.

A check is first made of the SEB port (or SDU port) to make sure that it is ready for transmission

(368) and the MCU 190 waits until the ready state is reached. If the SEB (or the SDU) port is ready for transmission, the associated internal buffer in the MCU 190 will be empty. If so, the MCU 190 sends a byte of data to either the SEB or the SDU(i.e., the interface controller 84). If the SEB (or SDU) port is not ready for transmission, then the routine enters a wait state until the port is ready for transmission. After a byte has been sent in block 370, a check is made to determine whether all bytes have been sent (373). If all bytes have not been sent, then the routine is exited so that the time shared processing may continue for other ports as shown in FIGURE 10B. If all bytes have been sent, as determined in block 372, then a status flag is reset (374). The setting of the status flag indicates that an additional message may be sent to the SEB (or the SDU) since the previous message has already been sent.

FIGURE 10F is a flowchart which represents the sequence of operations performed when data is received on a particular MCU 190 port. Like FIGURE 10E, the flowchart in FIGURE 10F is descriptive of the operations that are performed when information is received at both the SEB port and the SDU port. Additionally, the operations are descriptive of data reception at an SNES port. Initially, a check is made at block, 380 to determine whether any byte has been received at the SEB port (or SDU or SNES port). If a byte has been sent, then a check is made in block 382 to determine if the associated queue is full. If the check in block 382 reveals that the associated queue is not full, then the byte is retrieved and placed in the appropriate SEB, SDU, or SNS queue (384). If the associated queue is full or if no byte was received, then the routine exits at block 386.

Using the routines shown in FIGURES 10D, 10E, and 10F, the SEB, SDU and SNES ports serve as full duplex ports where data is both sent and received on each port. The SNES port is a parallel port and the SDU and SEB ports are serial ports.

FIGURES 10G and 10H are flowcharts which depict the sequence of operations performed when either SEB, SDU or SNES data is processed by the MCU 190. A check is initially made of the appropriate SEB, SDU, or SNES queue in MCU 190 to determine whether a byte has been received for processing. If so, one byte is taken from the appropriate queue (400). The routine shown in FIGURE 10G operates on one byte at a time.

After a byte is retrieved from the relevant queue, a check is made to determine whether the message is complete (462). Thus, if SEB data is being processed, a check is made to determine if the SEB message is complete. If SNES or SDU data is processed, then the check determines if the SNES or SDU message is completed. If the message is not complete, then the routine exits because the message is not ready to be processed. The next time the processing routine is performed, the next byte is retrieved until the complete message is obtained for processing.

If the message is complete, then a checksum is performed on the entirety of the message (404). If the checksum determination reveals an error, then a status flag is reset (406) and the message processing ceases and the routine is exited. If the checksum is correct, then a number of further checking operations are performed.

If the message is for the SEB (408) then, the message is sent to the SEB port (410). If the message is for the SDU, then the message is sent to the SDU port (410) and the routine exits.

If the check at block 408 yields a "NO" response, then a check is made at block 412 to determine whether the message is addressed to the SNES port. If the message is addressed to the SNES port, then block 414 is implemented by coupling the message to FIFO 184 (which is coupled to the SNES data lines) shown in FIGURE 6.

In processing SEB, SDU or SNES data, the routine then determines, if the response in block 412 is "NO", whether the command portion of the received message is a reset request (416). If a "reset" request has been received, then the MCU 190 resets itself (418). After the reset operation, the processing routine is exited.

If the check at block 416 reveals that a reset request has not been made, then in the case of SDU data processing only, a check is made to determine whether the command is a CPU pause request (419). If a pause request has been detected, then the CPU will be commanded to halt (421). If a CPU pause request is not detected, then in the case of SDU data, a check is made to determine if a CPU resume operation request has been made (423). If the command is a CPU resume request, in the case of SDU processing, the CPU will be resumed (425) and the routine will exit.

If the detected message command is not a CPU resume request, or if SEB or SNES data is being processed, then a check is made to determine whether a version ID request has been made as shown in block 420. If the version ID request has been made, then the version ID request is sent to either the SEB (if SEB data is being processed) or to the SDU (if SDU is being processed) (422). If SNES data is being processed, instead of checking whether a version ID request has been made, a check is made to determine whether the version ID request has been sent, and if so, the version ID is saved.

In the case of processing SEB data only, if block 420 indicates that a version ID request has not been made, a determination is made as to whether the data received is controller data (424). If controller data has been received, then as indicated at block 426, the data is sent by the MCU 190 to the latch 186 (in FIGURE 6) and to the SNES controller data lines. For

SEB data, the command portion of the received message indicates that the controller data is included in the message. Such processing is not done if SDU or SNES data is being processed by the MCU 190.

If the check in block 424 indicates that controller data has not been received, then a check is made to determine whether a status of various flag states have been requested (428). Status flags may indicate error conditions in high speed download operations or numerous other status conditions. If the check in block 428 indicates that a status message has been requested, then the status is sent to either the SEB, SDU or SNES depending upon whether SEB, SDU or SNES data is being processed (430) and the routine exits. If the check at block 428 reveals that a status flag state has not been requested, then the system determines that the unknown message is being processed and the processing routine is exited.

FIGURE 10I is a flowchart which delineates the sequence of operations if the MCU 190 detects an error in receiving data on one of its serial ports. If an error is detected, an internal interrupt routine causes branching to block 440 which determines whether there is an error in the byte received. If so, the receiver buffer is cleared (442) to delete the erroneous data. The flags which triggered the interrupt are then cleared (444) and the error routine is exited.

While the invention has been described in connection with what is presently considered to be the most practical and preferred embodiment, it is to be understood that the invention is not to be limited to the disclosed embodiment, but on the contrary, is intended to cover various modifications and equivalent arrangements included within the spirit and scope of the appended claims.

Claims

1. In an airline-based communication system having a master control computer including a storage device for storing a plurality of computer programs, a passenger seat display unit comprising:

 a memory system for receiving at least one of said plurality of programs

 a first data processor, coupled to said memory system for executing at least one of said plurality of programs stored in said storage device; and

 a second processor, coupled to said first data processor, for performing interface tasks for said first processor and for receiving said at least one of said plurality of programs to be executed by said first data processor, whereby said at least one of said plurality of programs may be loaded into said memory system.

2. A passenger seat display unit in accordance with claim 1, wherein said at least one of said plurality of programs is a video game program.

3. A passenger seat display unit in accordance with claim 1 further including a display for displaying videographics data resulting from the execution of said at least one of said plurality of programs.

4. A passenger seat display unit in accordance with claim 3, wherein said display is a liquid crystal display unit.

5. A passenger seat display unit in accordance with claim 1, further including a display, and a display interface processor coupled to said second processor, and said display.

6. A passenger seat display unit in accordance with claim 1, further including a tuner coupled to said second processor for receiving said at least one of said plurality of programs from said master control computer and for coupling said at least one of said plurality of programs to said second processor.

7. A passenger seat display unit in accordance with claim 6, further including a display coupled to said tuner, and an interface processor coupled to said tuner and said second processor.

8. A passenger seat display unit in accordance with claim 7, further including a card reader, coupled to said interface processor, for reading indicia embodied on a passenger's card.

9. A passenger seat display unit in accordance with claim 1, further including a display screen and display processing unit coupled to said first data processor for generating a video signal for displaying on said display screen.

10. A passenger seat display unit according to claim 1, further including an interface processor coupled to said second processor for providing data indicative of the type of program download operation which is to be attempted.

11. A passenger seat display unit according to claim 10, further including a display coupled to said interface processor.

12. An airplane based entertainment system comprising:

 a master control unit having a memory for storing a plurality of programs,

 a passenger seat based data processing unit associated with substantially every passenger seat on said airplane, each of said seat based

data processor units being coupled to said master control unit and including a storage device for receiving at least one of said plurality of programs from said memory and a processor for executing at least one of said plurality of programs.

13. An airplane based entertainment system in accordance with claim 12, wherein said at least one of said plurality of programs is a video game program.

14. An airplane based entertainment system in accordance with claim 12, wherein said passenger seat based data processing unit further includes a display for displaying videographics data resulting from the execution of said at least one of said plurality of programs.

15. An airplane based entertainment system in accordance with claim 14, wherein said display is a liquid crystal display unit.

16. An airplane based entertainment system in accordance with claim 14, wherein said passenger seat based data processing unit further includes a display interface processor coupled to said display.

17. An airplane based entertainment system in accordance with claim 12, wherein said passenger seat based data processing unit further includes a tuner coupled to said processor for receiving said at least one of said plurality of programs from said master control unit and for coupling said at least one of said plurality of programs to said processor.

18. An airplane based entertainment system in accordance with claim 17, further including a display coupled to said tuner, and an interface processor coupled to said tuner and said processor.

19. An airplane based entertainment system in accordance with claim 18, wherein said passenger seat display unit further includes a card reader, coupled to said interface processor, for reading indicia embodied on a passenger's card.

20. An airplane based entertainment system in accordance with claim 12 wherein said passenger seat based data processing unit further includes a display screen and display processing unit coupled to said processor for generating a video signal for display on said display screen.

21. An airplane based entertainment system in accordance with claim 12, wherein said passenger seat based data processing unit further includes a boot read-only memory for storing a program to be executed by said processor upon the power being turned on.

22. An airplane based entertainment system according to claim 12, wherein said passenger seat based data processing unit includes an interface control processor for receiving information to be coupled to said processor for processing.

23. An airplane based entertainment system according to claim 22 further including a display interface processor coupled to said interface control processor.

24. An airplane based entertainment system according to claim 12 wherein said seat based data processing unit includes a display interface processor coupled to said processor to provide data indicative of the type of program download operation which is to be attempted.

25. In an airplane based entertainment system having a master control unit and at least one seat display unit, a method of operating said airplane based entertainment system comprising the steps of:

downloading display menu generating software to said at least one seat display unit,

executing said display menu generating software at said seat display unit,

displaying a display menu at said seat display unit to permit the selection of a plurality of entertainment options; and

providing the selected entertainment in response to a passenger selection.

26. A method according to claim 25, wherein said downloading step includes the step of requesting information as to the type of download operation.

27. A method according to claim 26, further including the step of responding to the request for information about the type of download operation by an interface processor embodied within the seat display unit.

28. A method according to claim 25, wherein said downloading step includes the step of monitoring information that is downloaded and aborting the downloading operation if the information being monitored is not in a predetermined format.

29. A method according to claim 28, wherein said monitoring step includes the step of checking to determine whether the number of expected memory banks has been transmitted.

30. A method according to claim 28, wherein said monitoring step includes the step of checking to determine whether the expected starting address has been transmitted.

31. In an airplane based entertainment system having a plurality of seat display units, each having a data processor unit and a display, a method of operating said airplane based entertainment system comprising the steps of:

displaying at each of said seat display units a display menu identifying a plurality of entertainment options, and

providing to each of said plurality of seat display units the entertainment option selected by a passenger associated with said seat display unit.

32. A method according to claim 31 wherein said displaying step includes the step of displaying indicia indicative of different video games which may be selected for play at a seat display unit.

33. A method according to claim 32 further including the step of downloading a selected video game to a seat display unit at which said selected video game was selected.

34. A method according to claim 33, further including the step of executing said selected video game at the seat display unit at which the video game was selected.

35. A method according to claim 31, further including the step of providing a movie selection option for selecting movies to be viewed at said seat display unit.

36. A method according to claim 31 further including the step of providing a shopping option which may be selected.

37. A method according to claim 31 further including the step of providing a communications option which may be selected via said seat display unit.

38. A method according to claim 37 including the step of providing a facsimile option whcih may be selected.

39. A method according to claim 31 further including the step of providing a data processing option which may be selected.

40. A method according to claim 31 further including the step of providing a language option for selection.

41. In a data processing system having a memory system including at least one program memory and processing unit having a predetermined address space for executing programs stored in said at least one program memory, a memory system comprising:

a main program memory,

a plurality of address mapping registers,

a memory control circuit coupled to said plurality of address mapping registers for controlling the location of said main program memory in the address space of said processing unit dependent upon the contents of said plurality of address mapping registers.

42. A memory system according to claim 41, further including a scratch pad memory and a scratch pad memory control circuit coupled to at least one of said plurality of address mapping registers for controlling the location of said scratch pad memory in the address space of said processing unit dependent upon the contents of at least one of said plurality of registers.

43. A memory system according to claim 41, further including a nonvolatile memory and a nonvolatile memory control circuit coupled to at least one of said plurality of address mapping registers for controlling the location of said nonvolatile memory in the address space of said processing unit dependent upon the contents of at least one of said plurality of registers.

44. A memory system according to claim 41, further including a additional program memory and a additional program memory control circuit coupled to at least one of said plurality of address mapping registers for controlling the location of said additional program memory in the address space of said processing unit dependent upon the contents of at least one of said plurality of registers.

45. A memory system according to claim 44, wherein said additional program memory is a boot read-only memory.

46. A memory system according to claim 41, further including decoding logic coupled to each of said plurality of address mapping registers for receiving digital signals and for changing the contents of at least one said plurality of address mapping registers.

47. A memory system according to claim 46, wherein said processing unit includes an address bus and wherein said decoding logic receives said digital signals on said address bus.

48. A memory system according to claim 41, further including an interface processing device for coupling information to said processor unit, and for receiving program information to be loaded into said main program memory for execution by said processor unit.

49. A memory system according to claim 48, further including at least one data bus coupled to said processor unit, at least one one buffer memory coupled to said interface processing device and coupled to said at least one data bus.

50. A memory system according to claim 48, further including an address bus coupled to said processor unit and control logic coupled to said address bus and said interface processing device said control logic controlling in part input/output operations of said interface processing device in response to signals received on said address bus.

51. A memory system according to claim 50, wherein said processor unit includes at least one data bus and control logic includes a plurality latches coupled to said interface processing device for receiving information for or coupling information to said at least one data bus.

52. A memory system according to claim 48, further including halt signal generating logic coupled to said interface processing device and said processor unit for coupling a halt control signal to said processor unit in response to a signal from said interface processing device.

53. In a distributed processing multiprocessor system having a control computer and at least one user interactive processing system, coupled to said control computer, a method of downloading information to said user interactive processing system comprising the step of:

requesting by the user interactive processing system a download operation from the control computer,

receiving at least one byte of downloaded data,

interpreting said at least one byte of data as defining predetermined configuration data; and

aborting said downloading operation if said predetermined configuration data is not received in the manner expected.

54. A method according to claim 53, wherein said predetermined configuration data is the number of memory banks expected.

55. A method according to claim 53, wherein said configuration data is the starting address expected.

56. A method according to claim 53, wherein said configuration data is the number of bytes expected.

57. A method according to claim 53, wherein said step of requesting includes the step of requesting a predetermined type of download.

58. A method according to claim 57, wherein said request is responded to by an interface processor embodied in said user interactive processing system.

FIG. 1

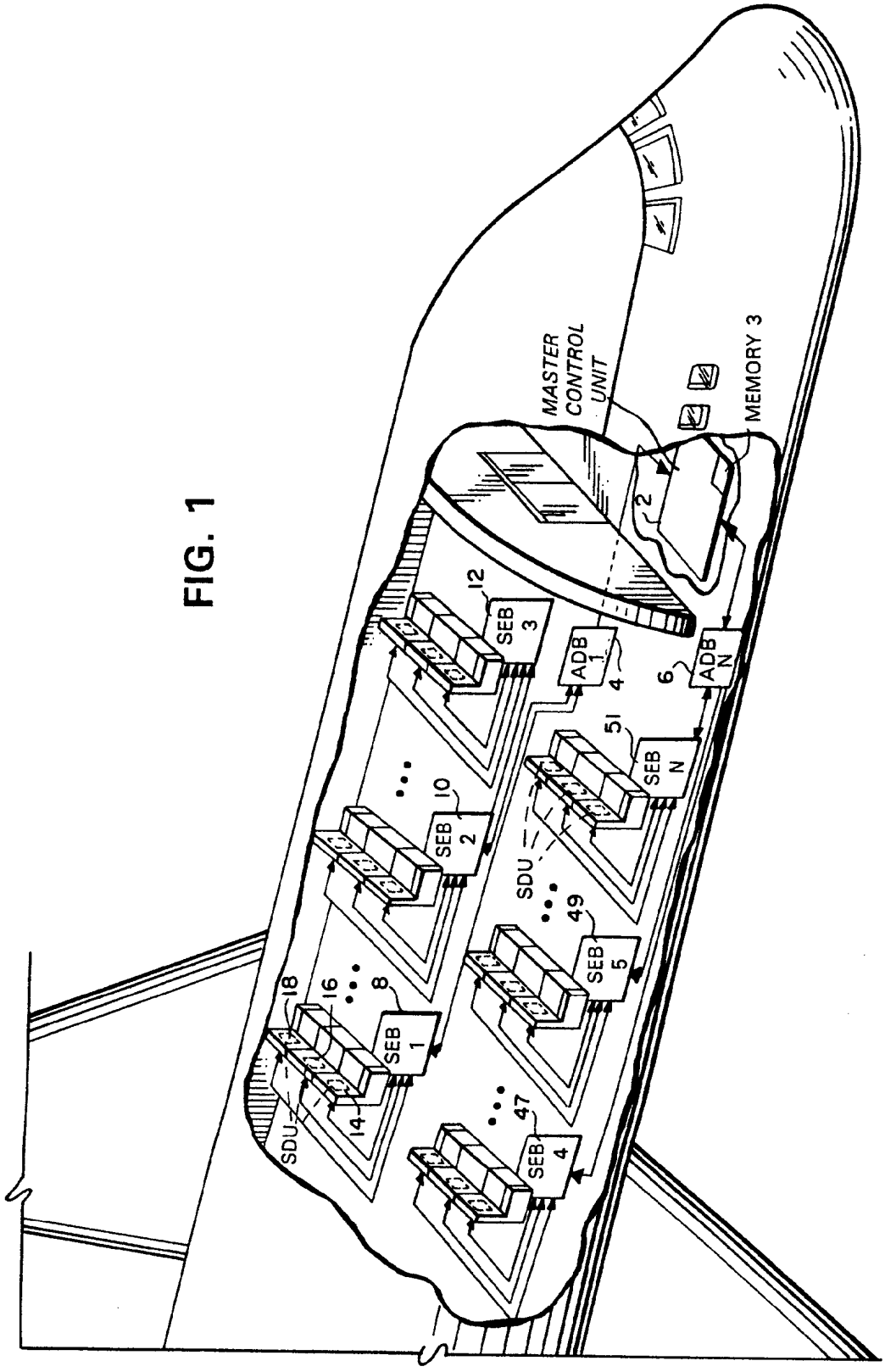
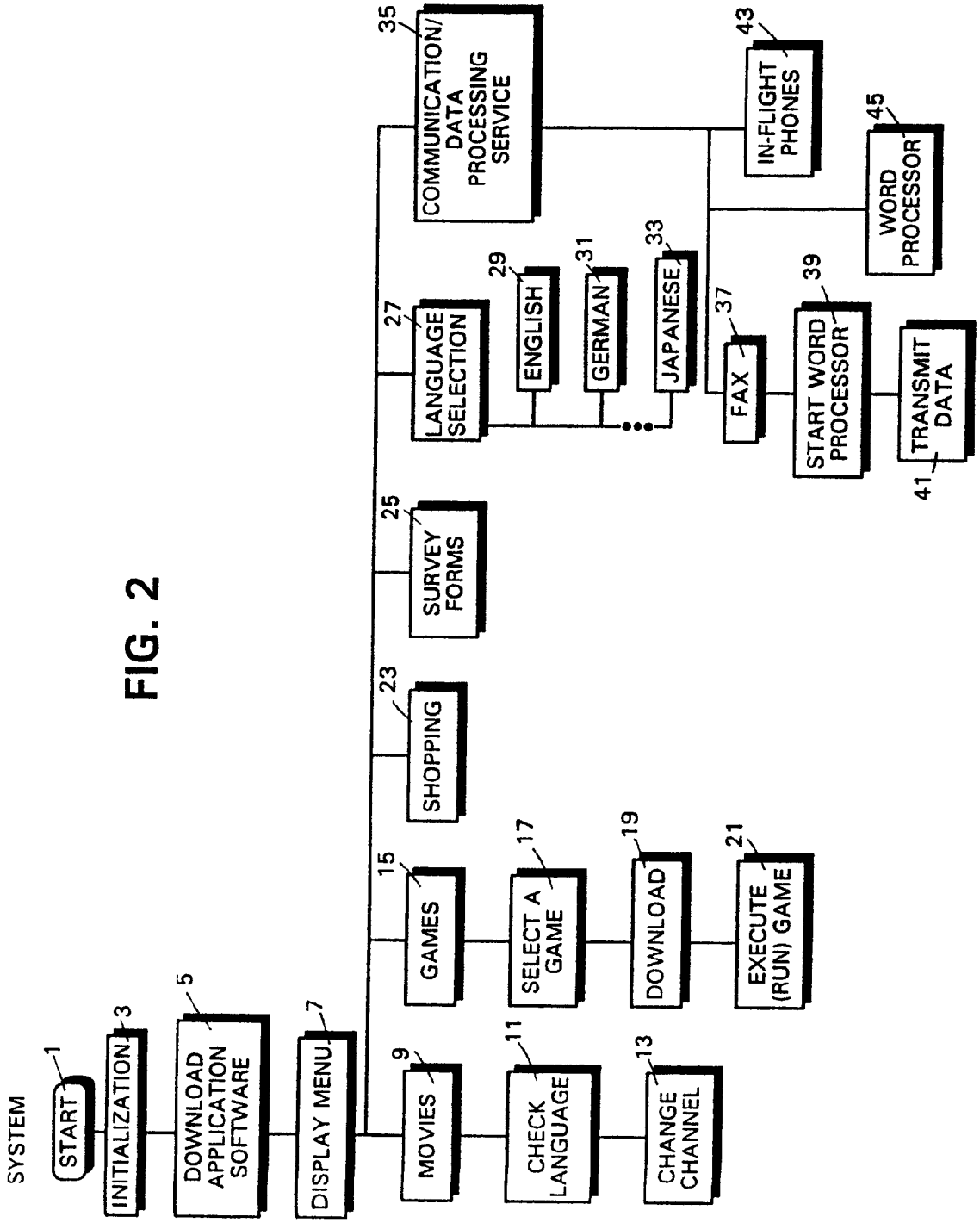

MASTER CONTROL UNIT
MEMORY 3
2
ADB 1
4
ADB N
6
SEB 1
8
SEB 2
10
SEB 3
12
SEB 4
47
SEB 5
49
SEB N
51
SDU
14
16
18

FIG. 2


SYSTEM
START
1
INITIALIZATION
3
DOWNLOAD APPLICATION SOFTWARE
5
DISPLAY MENU
7
MOVIES
9
CHECK LANGUAGE
11
CHANGE CHANNEL
13
GAMES
15
SELECT A GAME
17
DOWNLOAD
19
EXECUTE (RUN) GAME
21
SHOPPING
23
SURVEY FORMS
25
LANGUAGE SELECTION
27
ENGLISH
29
GERMAN
31
JAPANESE
33
COMMUNICATION/ DATA PROCESSING SERVICE
35
FAX
37
START WORD PROCESSOR
39
TRANSMIT DATA
41
IN-FLIGHT PHONES
43
WORD PROCESSOR
45

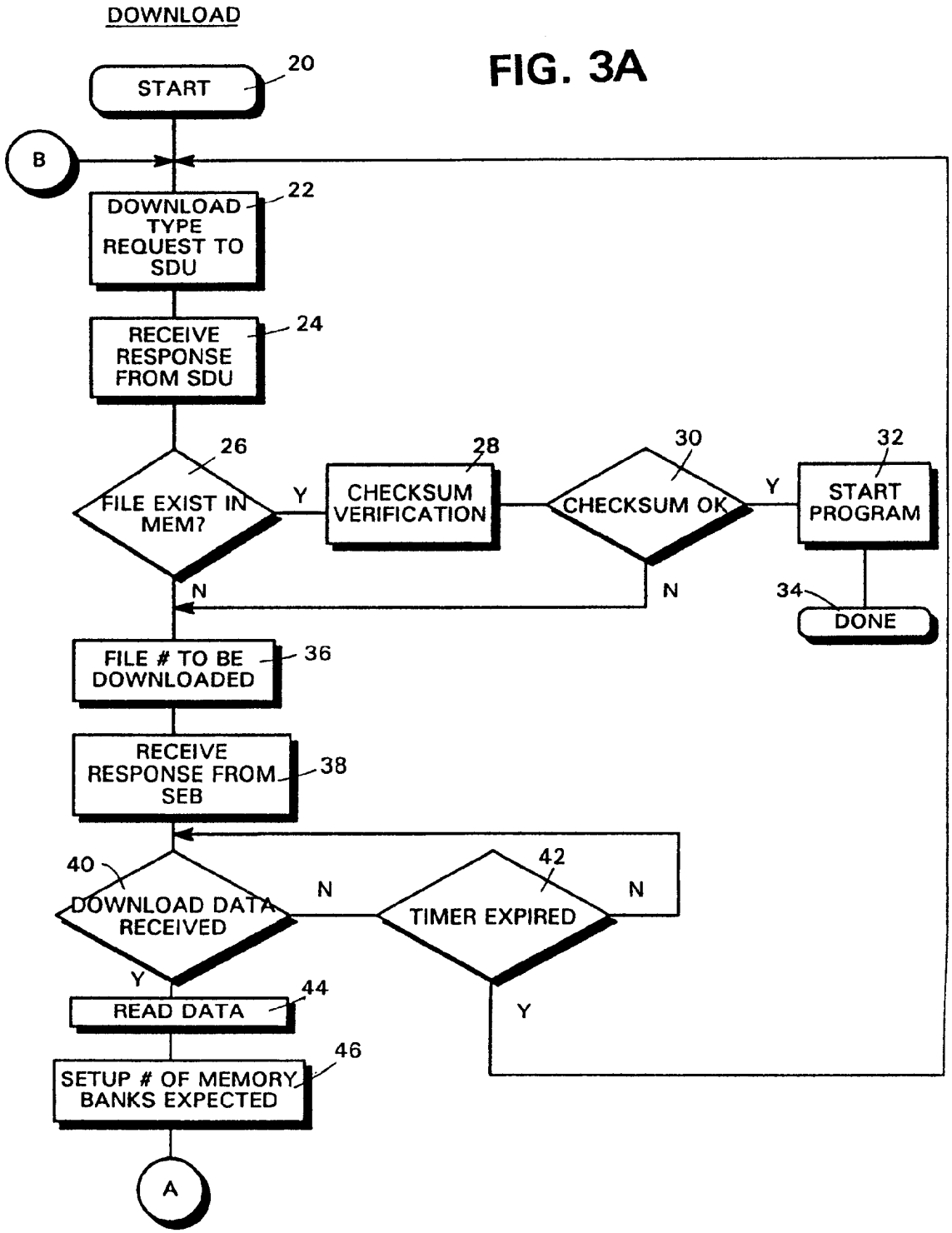

DOWNLOAD
START
20
B
DOWNLOAD TYPE REQUEST TO SDU
22
RECEIVE RESPONSE FROM SDU
24
FILE EXIST IN MEM?
26
Y
CHECKSUM VERIFICATION
28
CHECKSUM OK
30
Y
START PROGRAM
32
N
N
34
DONE
FILE # TO BE DOWNLOADED
36
RECEIVE RESPONSE FROM SEB
38
40
DOWNLOAD DATA RECEIVED
N
TIMER EXPIRED
42
N
Y
Y
READ DATA
44
SETUP # OF MEMORY BANKS EXPECTED
46
A

FIG. 3A

FIG. 3B

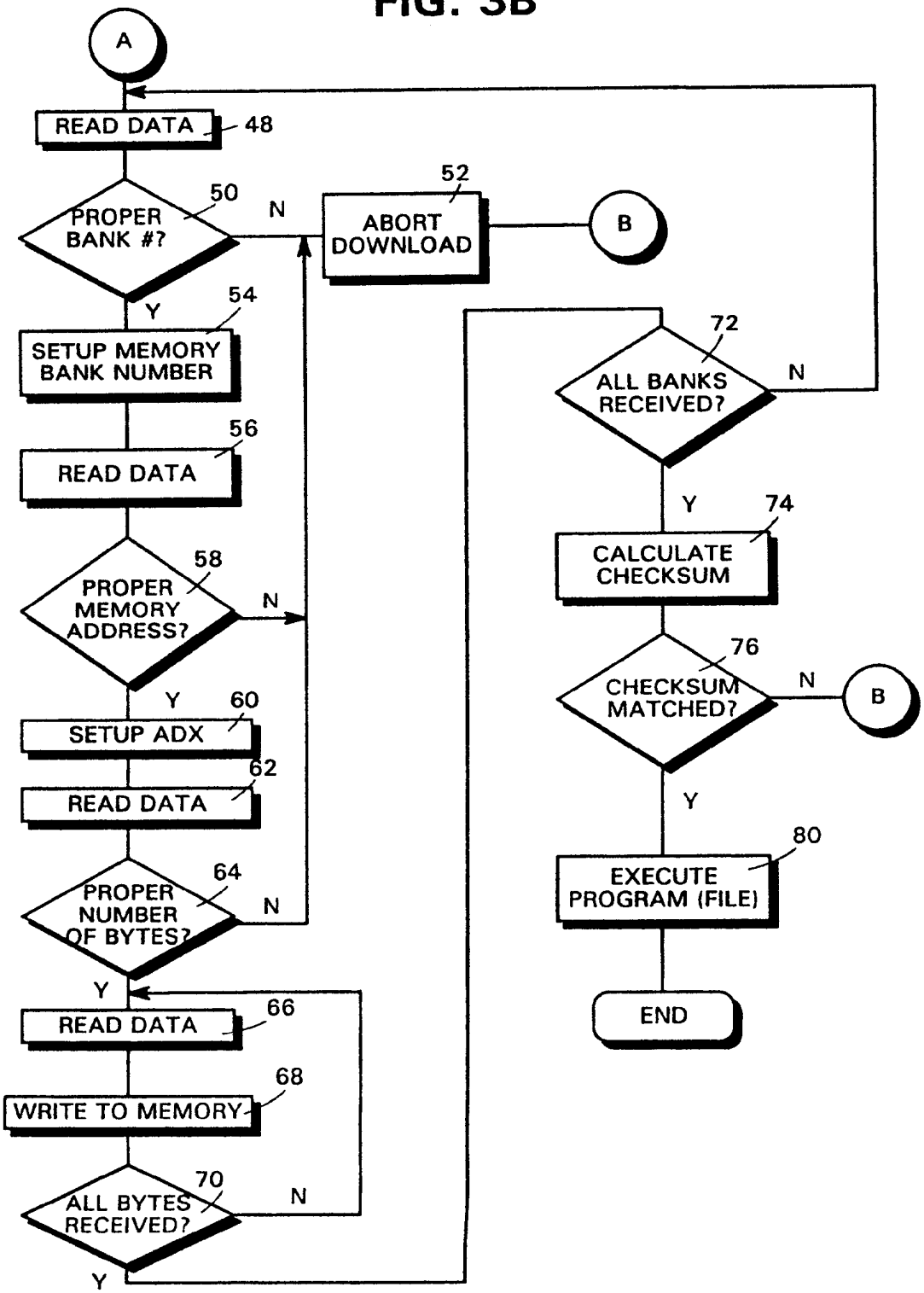

A
READ DATA
48
PROPER BANK #?
50
N
52
ABORT DOWNLOAD
B
Y
54
SETUP MEMORY BANK NUMBER
56
READ DATA
PROPER MEMORY ADDRESS?
58
N
Y
60
SETUP ADX
62
READ DATA
PROPER NUMBER OF BYTES?
64
N
Y
READ DATA
66
68
WRITE TO MEMORY
ALL BYTES RECEIVED?
70
N
Y
72
ALL BANKS RECEIVED?
N
Y
74
CALCULATE CHECKSUM
76
CHECKSUM MATCHED?
N
B
Y
80
EXECUTE PROGRAM (FILE)
END

FIG. 4

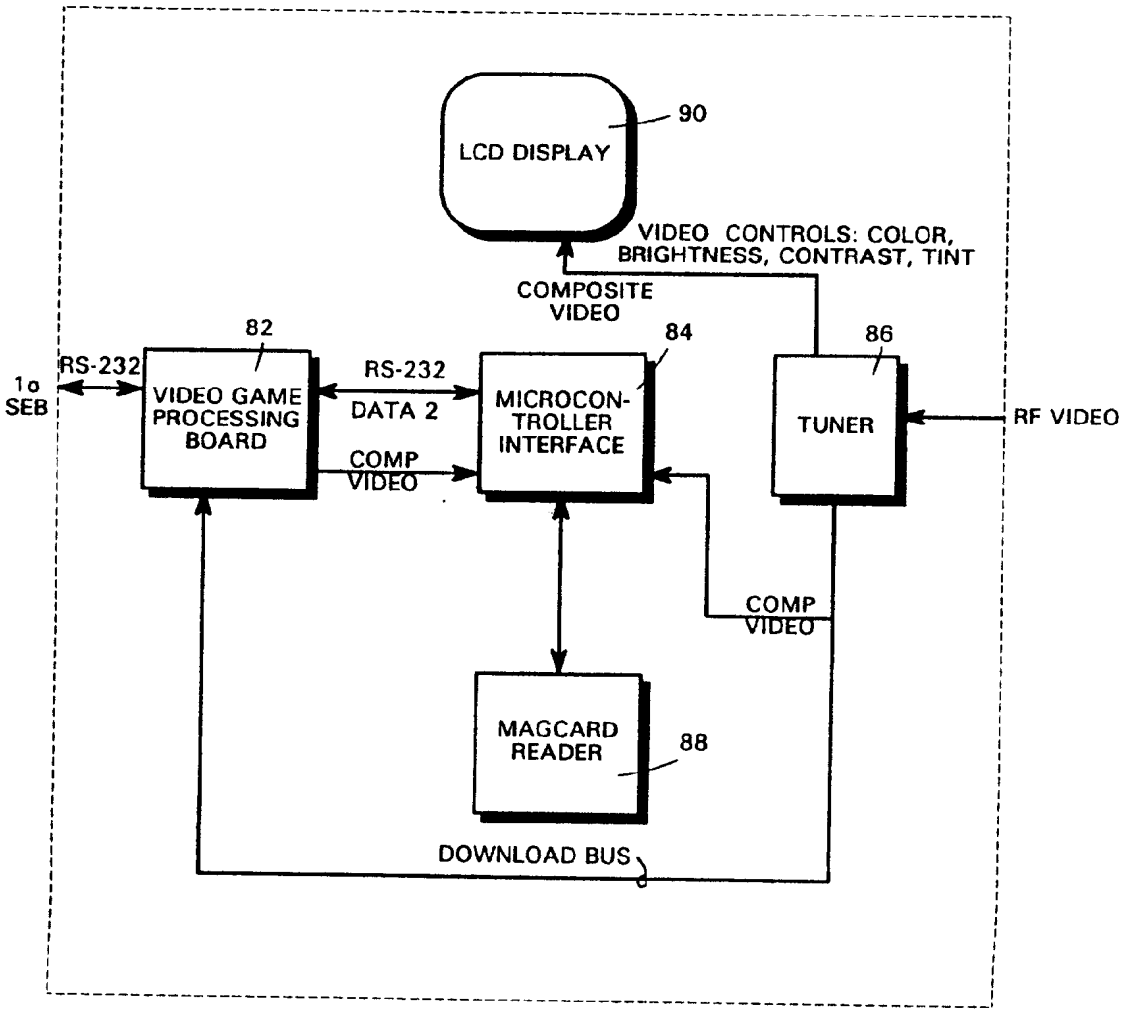
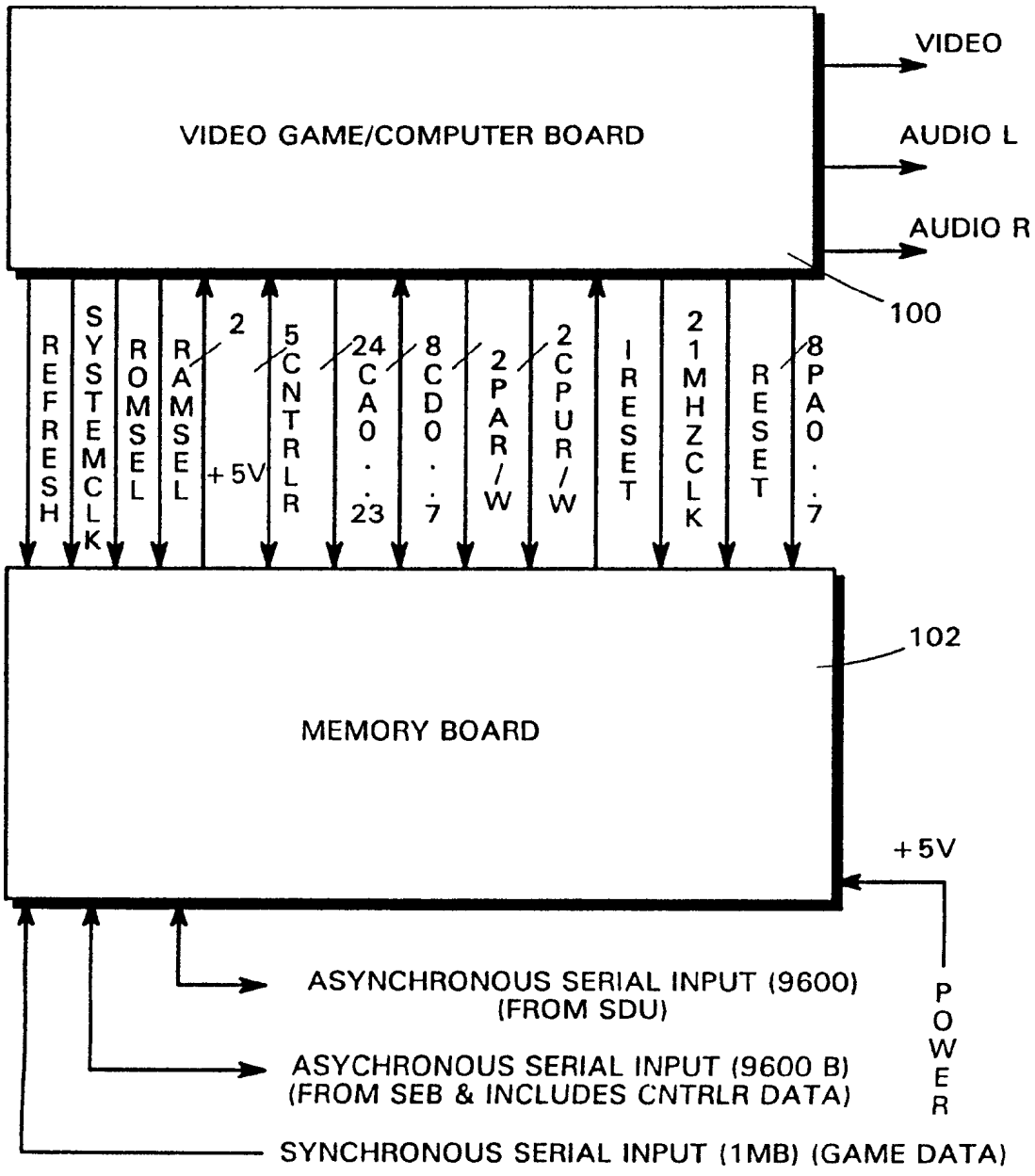

LCD DISPLAY
90
VIDEO CONTROLS: COLOR, BRIGHTNESS, CONTRAST, TINT
COMPOSITE VIDEO
82
84
86
RS-232
1o SEB
VIDEO GAME PROCESSING BOARD
RS-232
DATA 2
COMP VIDEO
MICROCON-TROLLER INTERFACE
TUNER
RF VIDEO
COMP VIDEO
MAGCARD READER
88
DOWNLOAD BUS

FIG. 5


VIDEO GAME/COMPUTER BOARD
VIDEO
AUDIO L
AUDIO R
100
REFRESH
SYSTEMCLK
ROMSEL
RAMSEL
2
+5V
5
CNTRLR
24
CA0 . . 23
8
CD0 . . 7
2
PAR/W
2
CPUR/W
IRESET
21MHZCLK
RESET
8
PA0 . . 7
MEMORY BOARD
102
+5V
ASYNCHRONOUS SERIAL INPUT (9600)
(FROM SDU)
ASYCHRONOUS SERIAL INPUT (9600 B)
(FROM SEB & INCLUDES CNTRLR DATA)
SYNCHRONOUS SERIAL INPUT (1MB) (GAME DATA)
POWER

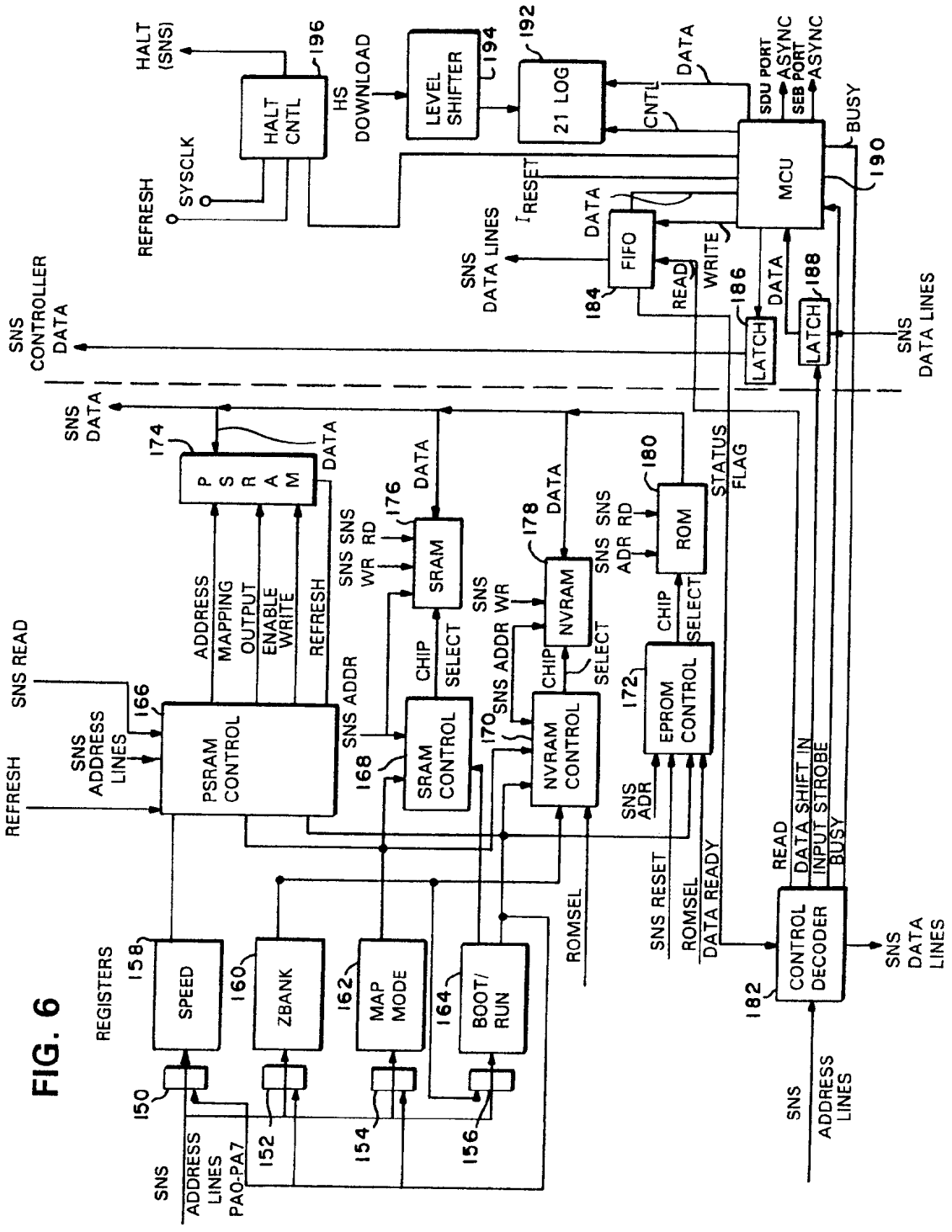
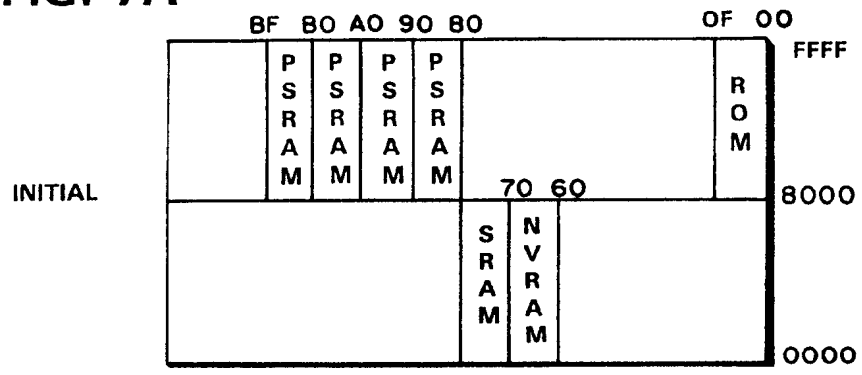

HALT (SNS)
196
HALT CNTL
HS DOWNLOAD
LEVEL SHIFTER
194
192
21 LOG
CNTL
DATA
SDU PORT
ASYNC
SEB PORT
ASYNC
BUSY
190
MCU
REFRESH
SYSCLK
RESET
DATA
FIFO
READ
WRITE
186
DATA
188
LATCH
LATCH
SNS DATA LINES
SNS CONTROLLER DATA
SNS DATA
SNS DATA LINES
174
PSRAM
DATA
ADDRESS MAPPING
OUTPUT ENABLE
WRITE REFRESH
SNS ADDR
SNS WR
SNS RD
176
SRAM
DATA
CHIP SELECT
178
NVRAM
DATA
SNS WR
SNS ADDR
CHIP SELECT
180
ROM
SNS ADR RD
STATUS FLAG
CHIP SELECT
172
EPROM CONTROL
SNS ADR
SNS READ
166
SNS ADDRESS LINES
REFRESH
PSRAM CONTROL
168
SRAM CONTROL
170
NVRAM CONTROL
ROMSEL
SNS RESET
ROMSEL
DATA READY
READ
DATA SHIFT IN
INPUT STROBE
BUSY
182
CONTROL DECODER
SNS DATA LINES
SNS ADDRESS LINES
REGISTERS
158
SPEED
160
ZBANK
162
MAP MODE
164
BOOT/ RUN
150
152
154
156
SNS ADDRESS LINES PA0-PA7

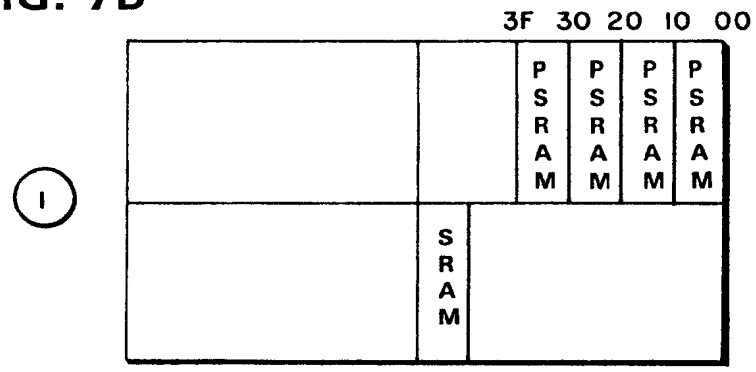
FIG. 6

FIG. 7A



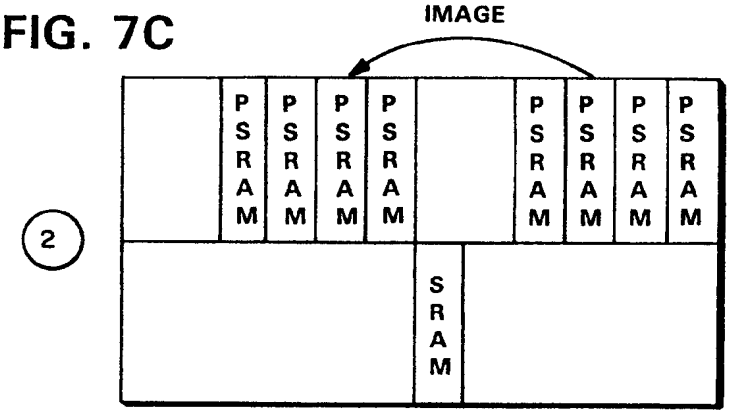
MAP MODE = 0
BOOT/RUN = 0
SPEED = X
ZBANK = X

FIG. 7B



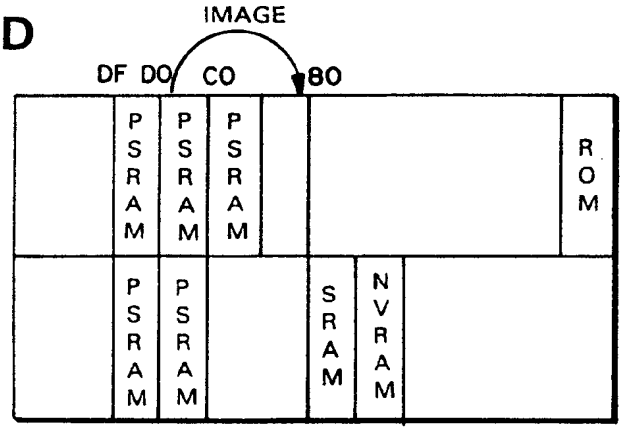
MAP MODE = 0
BOOT/RUN = 1
SPEED = 0
ZBANK = 0

FIG. 7C



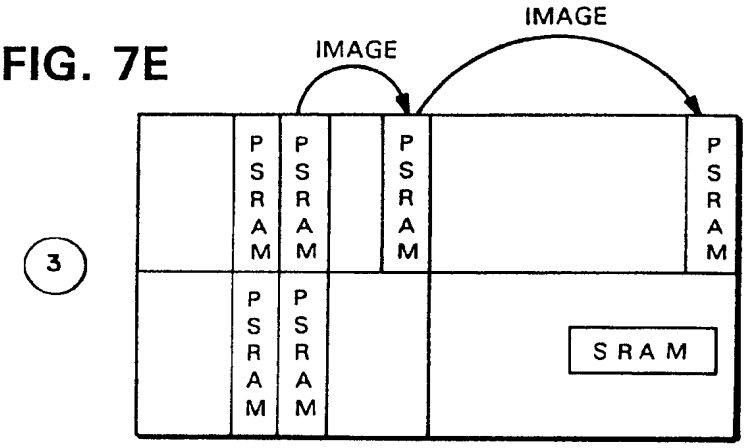
MAP MODE = 0
BOOT/RUN = 1
SPEED = 1
ZBANK = 0

FIG. 7D



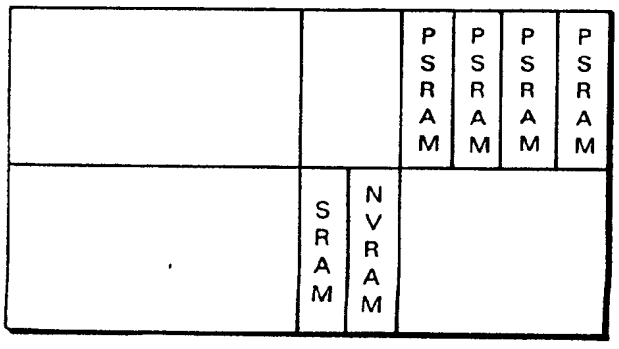
MAP MODE = 1
BOOT/RUN = 0
SPEED = X
ZBANK = X

FIG. 7E



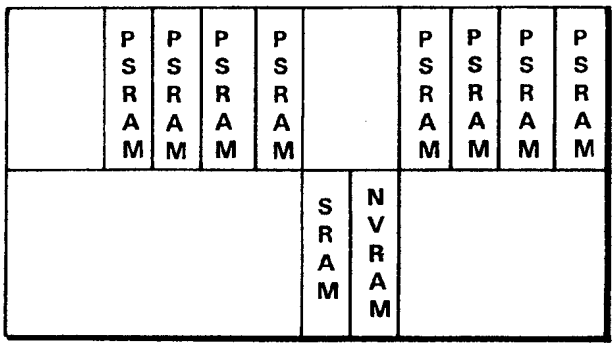
MAP MODE = 1
BOOT/RUN = 1
SPEED = X
ZBANK = 0

FIG. 7F



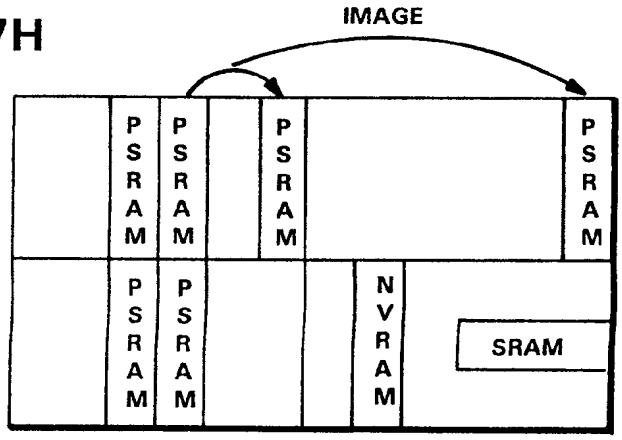
MAP MODE = 0
BOOT/RUN = 1
SPEED = 0
ZBANK = 1

FIG. 7G



MAP MODE = 0
BOOT/RUN = 1
SPEED = 1
ZBANK = 1

FIG. 7H



MAP MODE = 1
BOOT/RUN = 1
SPEED = 1
ZBANK = 1

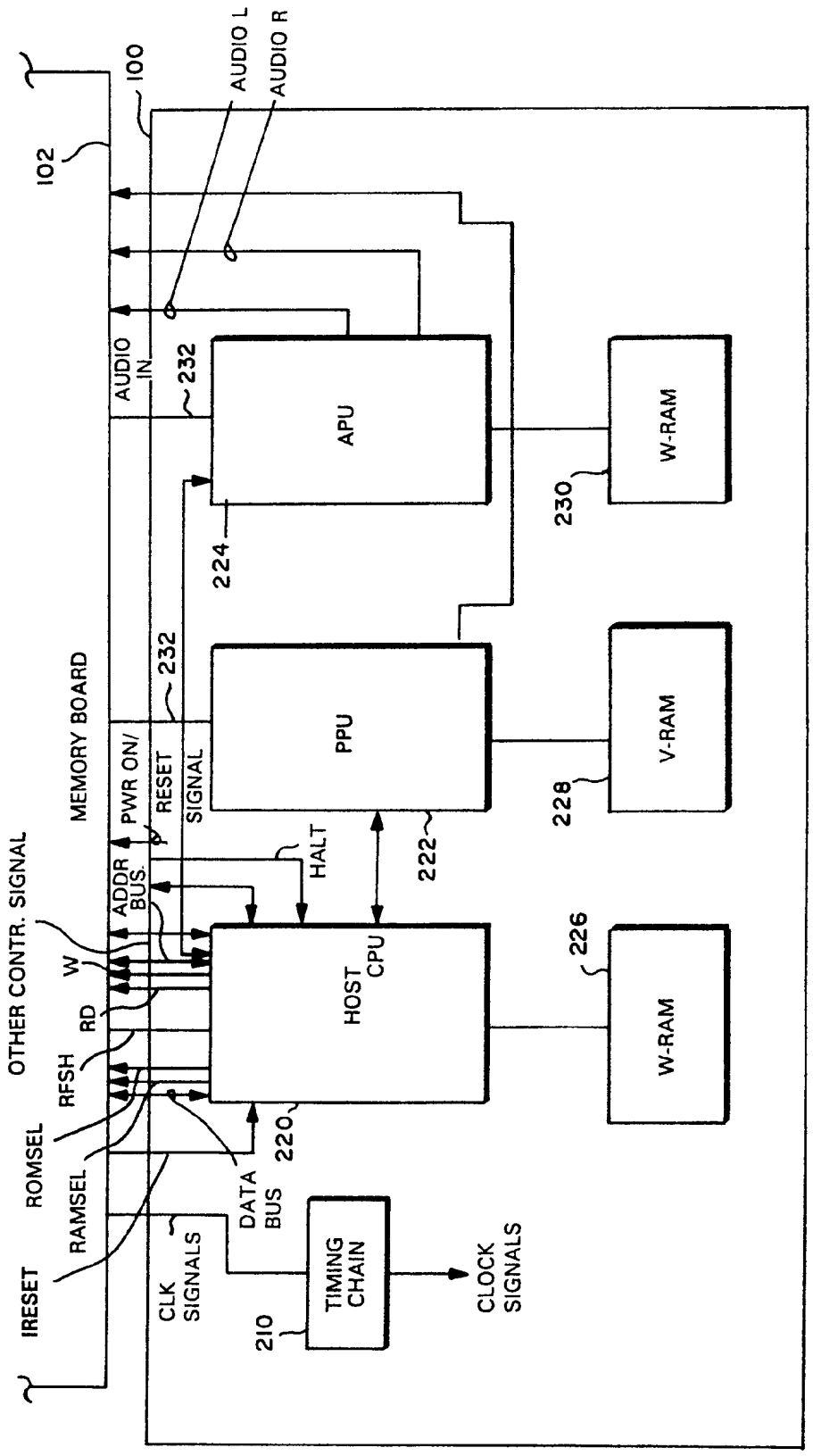
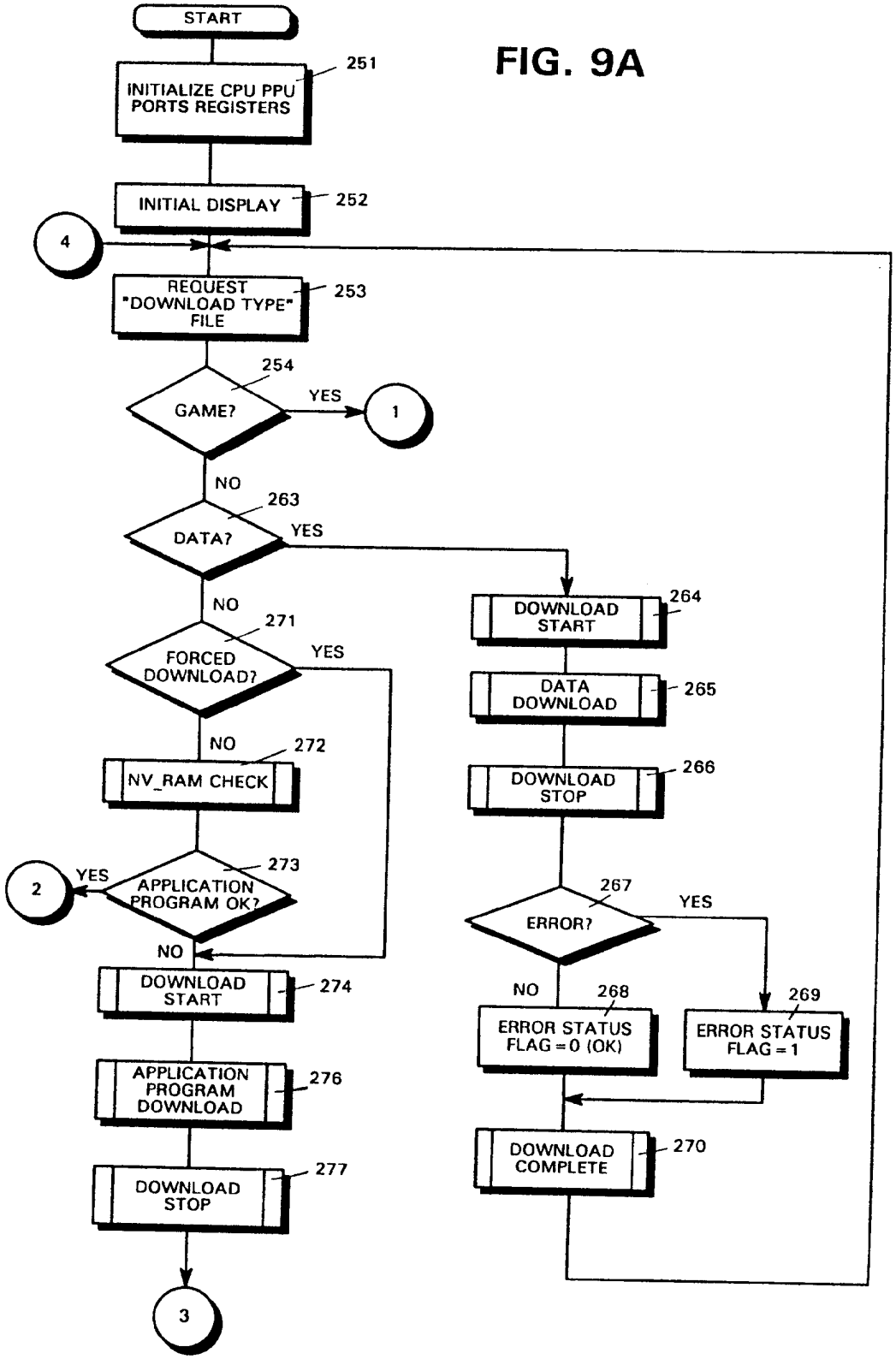

100
102
AUDIO L
AUDIO R
AUDIO IN
232
APU
224
230
W-RAM
MEMORY BOARD
PWR ON/ RESET SIGNAL
232
PPU
228
V-RAM
HALT
222
ADDR BUS.
OTHER CONTR. SIGNAL
W
RD
RFSH
ROMSEL
RAMSEL
IRESET
HOST CPU
226
W-RAM
220
DATA BUS
CLK SIGNALS
TIMING CHAIN
210
CLOCK SIGNALS

FIG. 8

FIG. 9A


START
INITIALIZE CPU PPU PORTS REGISTERS
251
INITIAL DISPLAY
252
4
REQUEST "DOWNLOAD TYPE" FILE
253
254
GAME?
YES
1
NO
263
DATA?
YES
NO
271
FORCED DOWNLOAD?
YES
NO
272
NV_RAM CHECK
273
2
YES
APPLICATION PROGRAM OK?
NO
DOWNLOAD START
274
APPLICATION PROGRAM DOWNLOAD
276
DOWNLOAD STOP
277
3
DOWNLOAD START
264
DATA DOWNLOAD
265
DOWNLOAD STOP
266
267
ERROR?
YES
NO
268
ERROR STATUS FLAG = 0 (OK)
269
ERROR STATUS FLAG = 1
DOWNLOAD COMPLETE
270

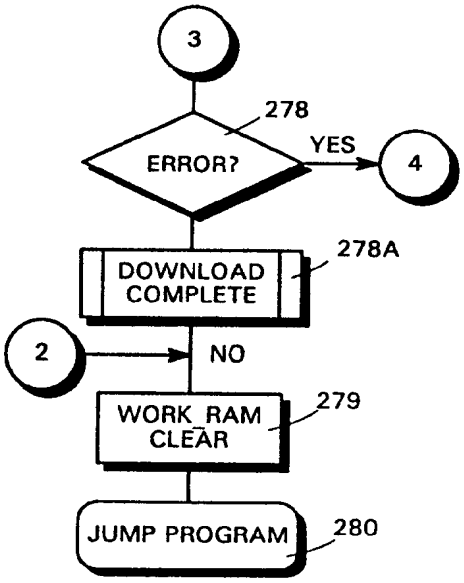

3
278
ERROR?
YES
4
DOWNLOAD COMPLETE
278A
2
NO
WORK_RAM CLEAR
279
JUMP PROGRAM
280

FIG. 9B

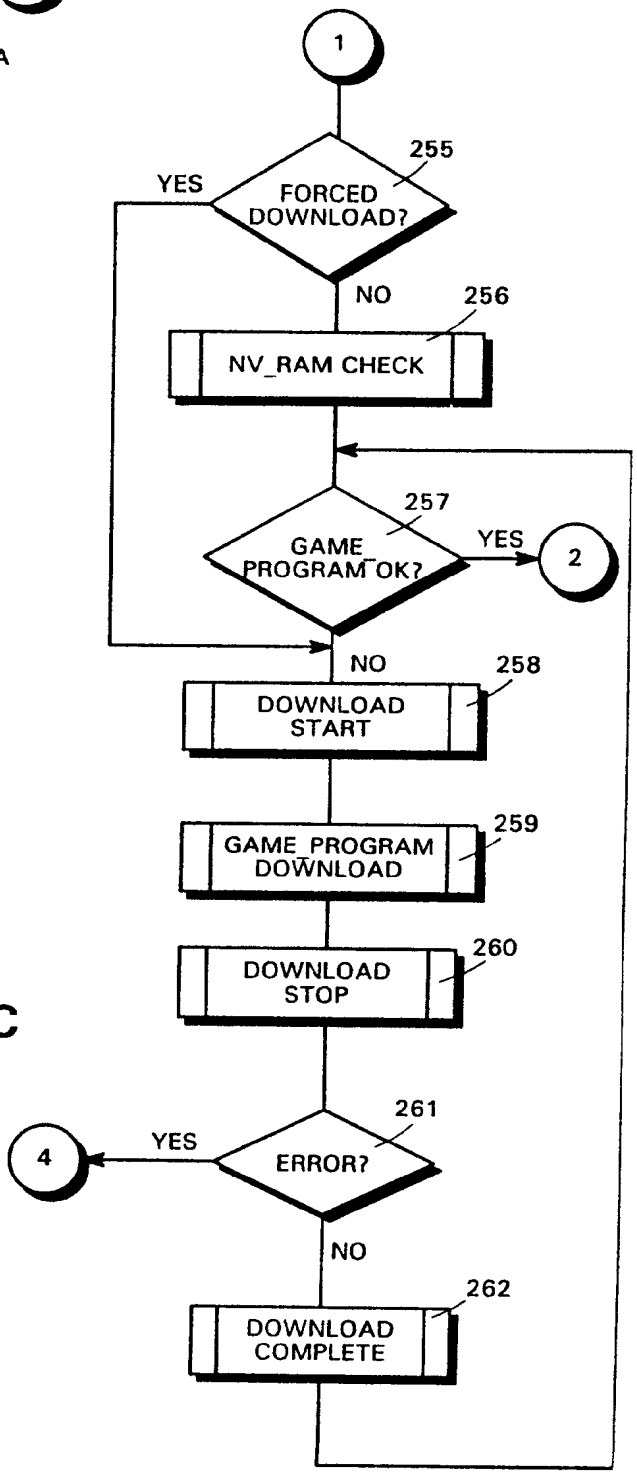
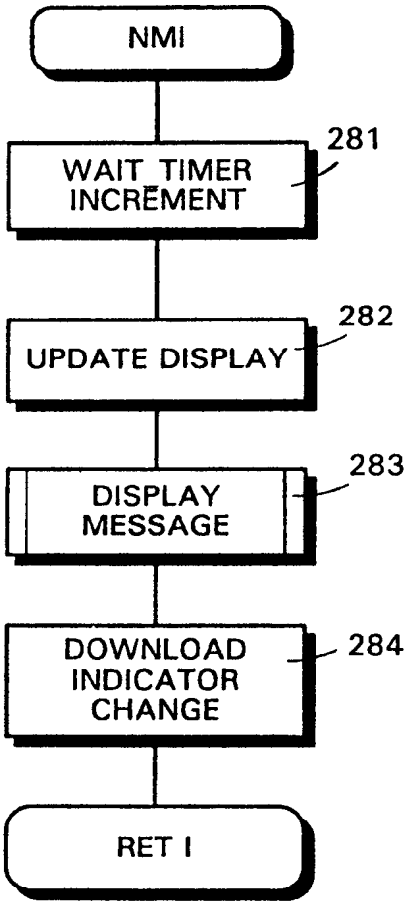

1
255
YES
FORCED DOWNLOAD?
NO
256
NV_RAM CHECK
257
GAME_PROGRAM OK?
YES
2
NO
258
DOWNLOAD START
259
GAME_PROGRAM DOWNLOAD
260
DOWNLOAD STOP
261
4
YES
ERROR?
NO
262
DOWNLOAD COMPLETE

FIG. 9C

FIG. 9D


NMI
WAIT TIMER INCREMENT
281
UPDATE DISPLAY
282
DISPLAY MESSAGE
283
DOWNLOAD INDICATOR CHANGE
284
RET I

MAIN PROGRAM

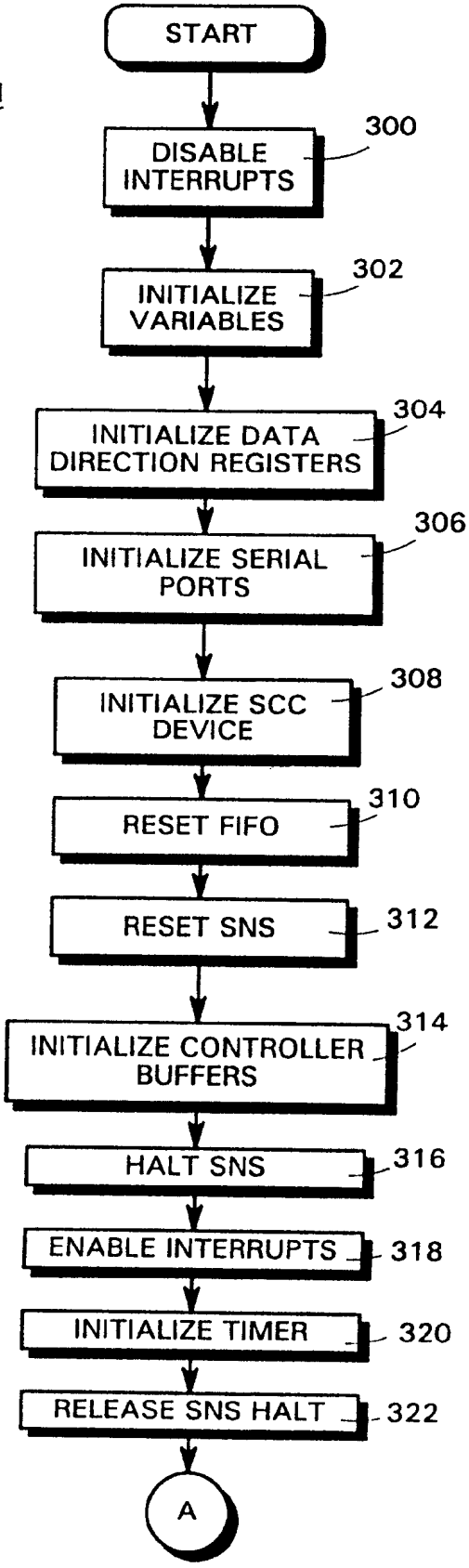


FIG. 10A

FIG. 10B

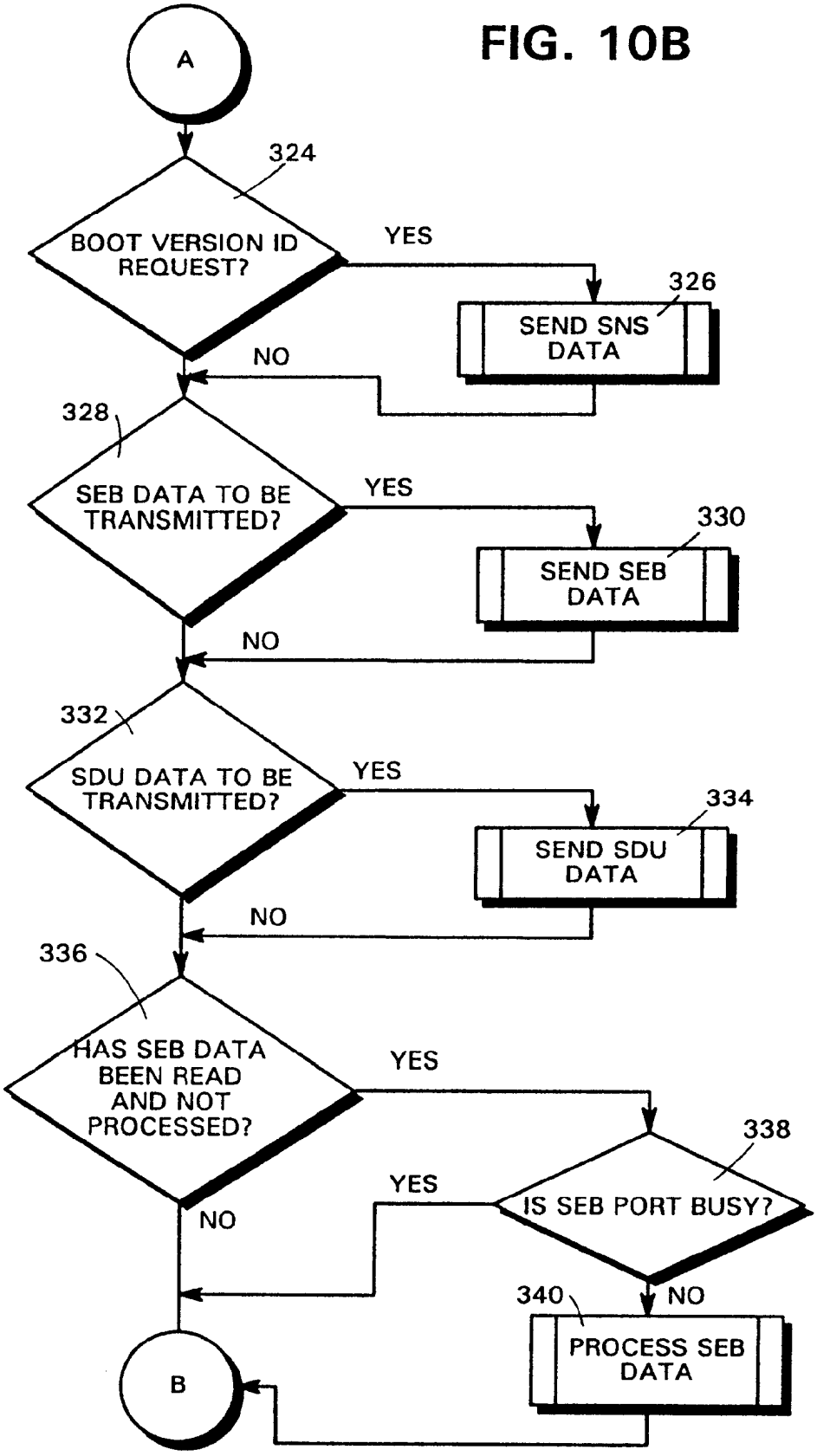

A
324
BOOT VERSION ID REQUEST?
YES
326
SEND SNS DATA
NO
328
SEB DATA TO BE TRANSMITTED?
YES
330
SEND SEB DATA
NO
332
SDU DATA TO BE TRANSMITTED?
YES
334
SEND SDU DATA
NO
336
HAS SEB DATA BEEN READ AND NOT PROCESSED?
YES
338
IS SEB PORT BUSY?
YES
NO
340
PROCESS SEB DATA
NO
B

FIG. 10C

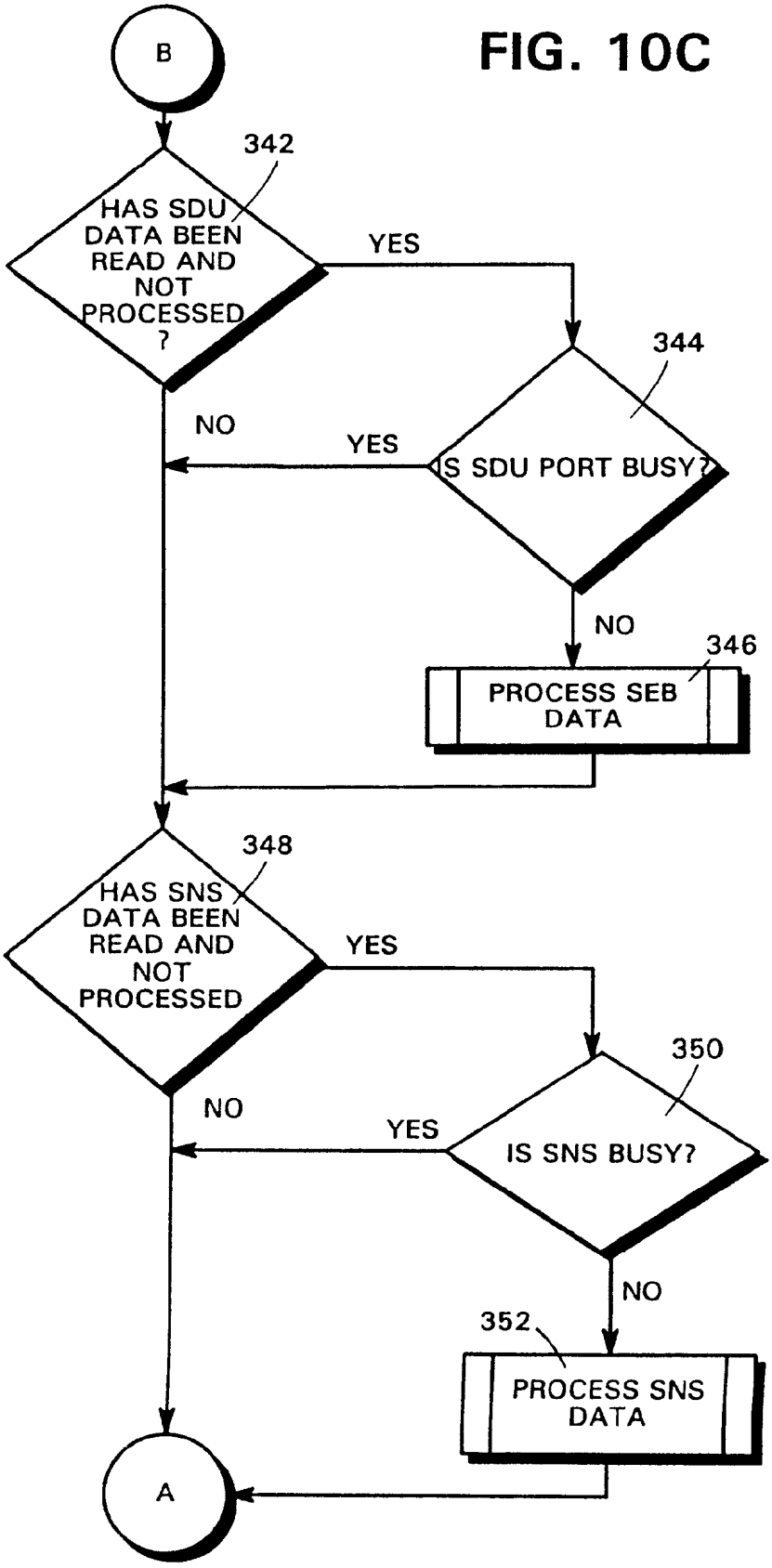
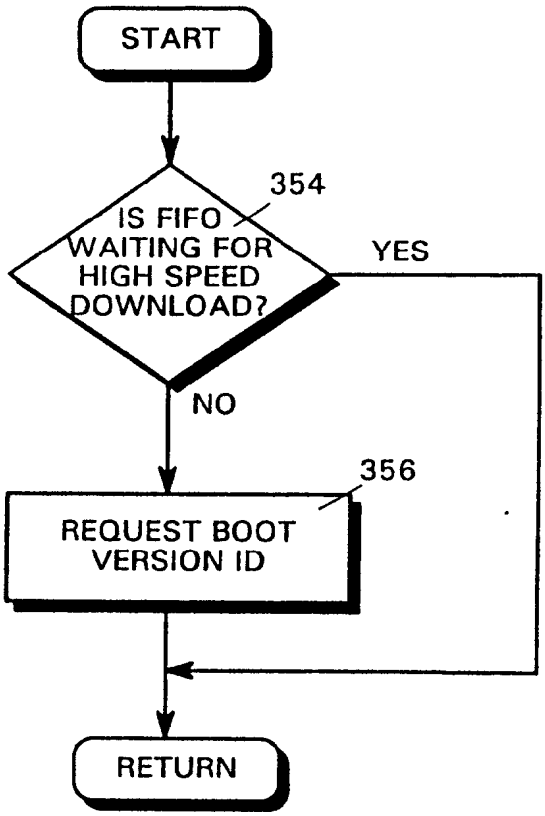

B
342
HAS SDU DATA BEEN READ AND NOT PROCESSED ?
YES
344
NO
YES
IS SDU PORT BUSY?
NO
346
PROCESS SEB DATA
348
HAS SNS DATA BEEN READ AND NOT PROCESSED
YES
350
NO
YES
IS SNS BUSY?
NO
352
PROCESS SNS DATA
A

FIG. 10D

SEND SNS DATA



SEND SEB (OR SDU) DATA

FIG. 10E

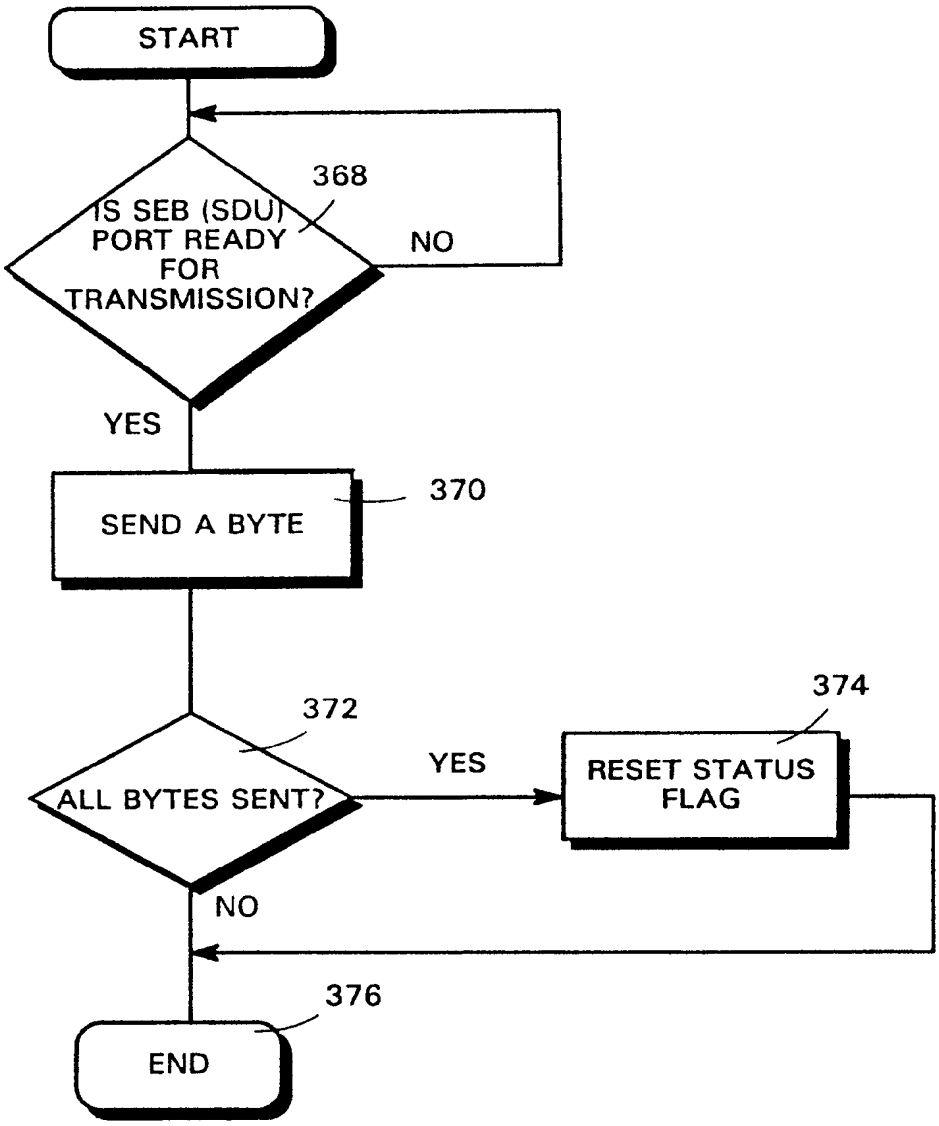
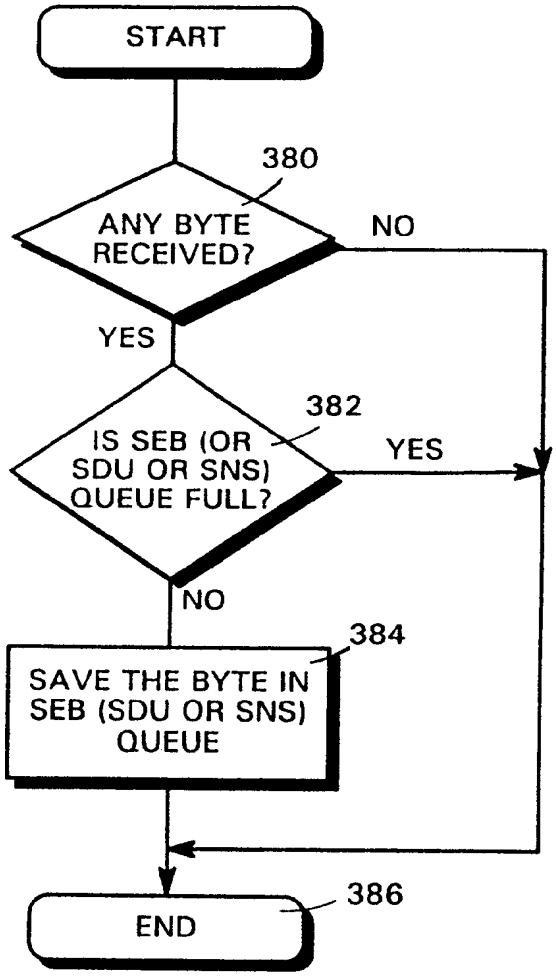


FIG. 10F

SERIAL PORT RECEIVER - SEB (SDU OR SNS) PORT



PROCESS SEB (SDU, SNS) DATA

FIG. 10G

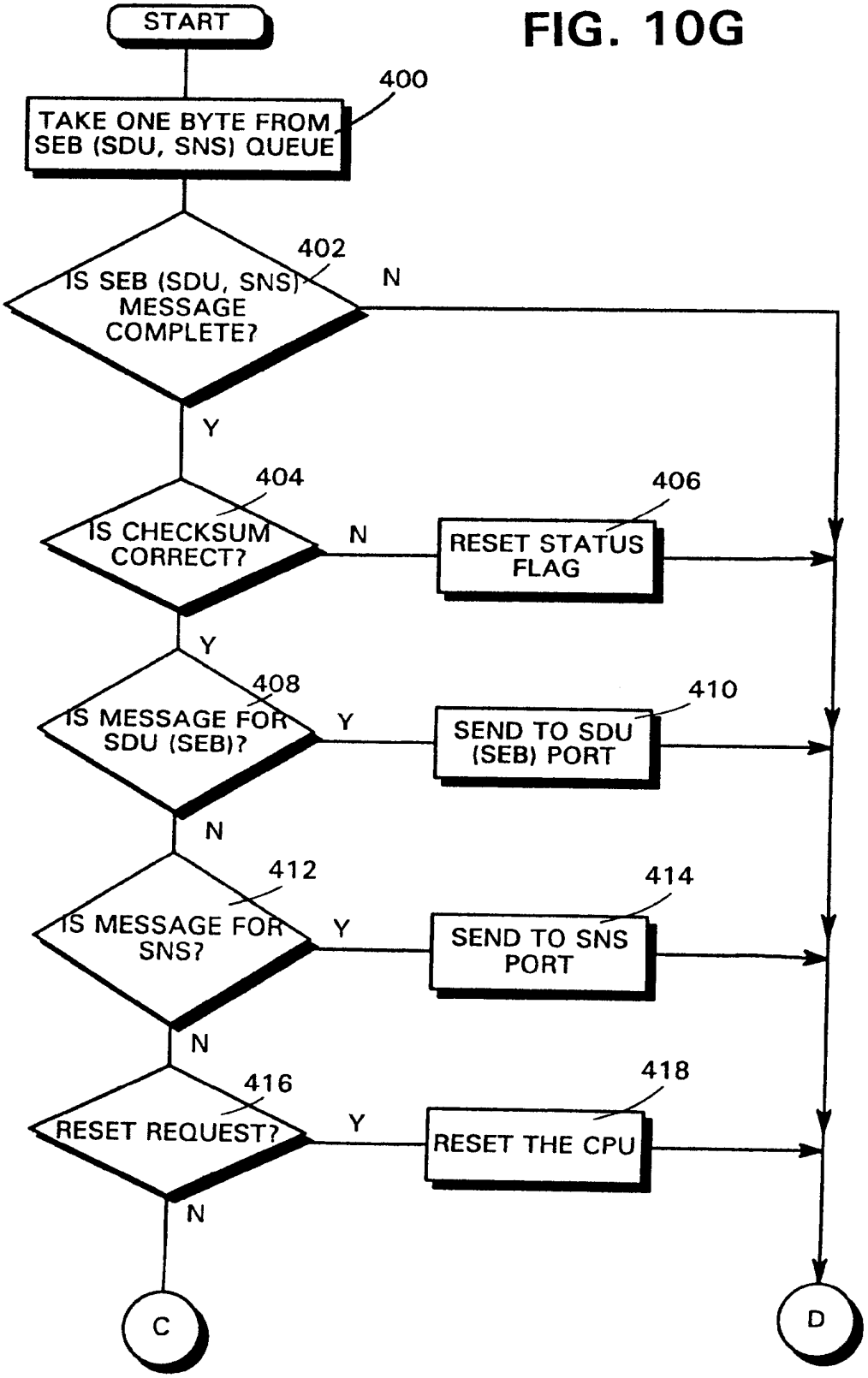


FIG. 10H

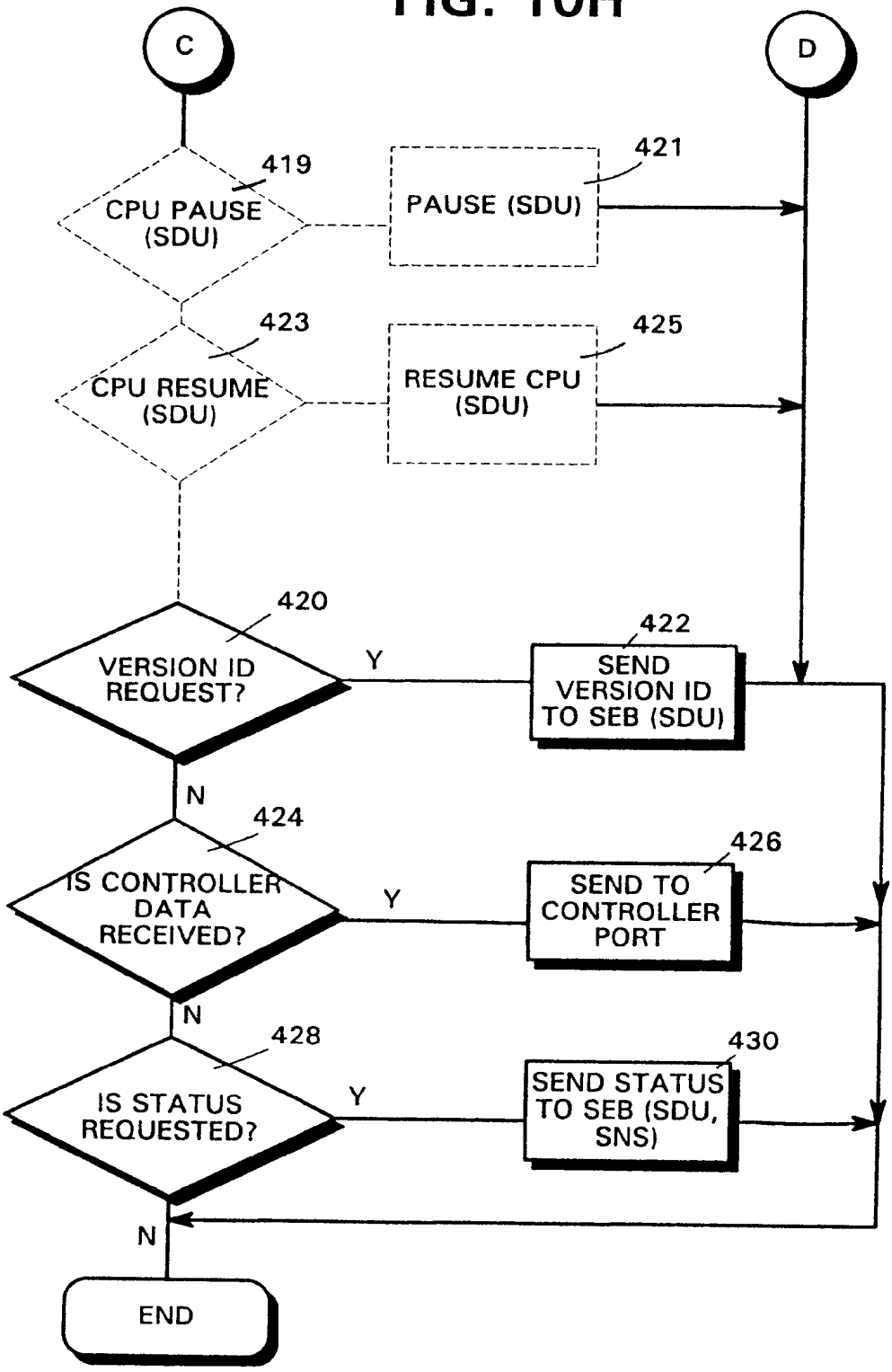

C
D
419
CPU PAUSE (SDU)
421
PAUSE (SDU)
423
CPU RESUME (SDU)
425
RESUME CPU (SDU)
420
VERSION ID REQUEST?
Y
422
SEND VERSION ID TO SEB (SDU)
N
424
IS CONTROLLER DATA RECEIVED?
Y
426
SEND TO CONTROLLER PORT
N
428
IS STATUS REQUESTED?
Y
430
SEND STATUS TO SEB (SDU, SNS)
N
END

FIG. 10I

ERROR IN RECEIVER

